Journal of Computer and System Sciences 78 (2012) 1045–1098

Contents lists available at SciVerse ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

The implication problem for 'closest node' functional dependencies in complete XML documents

M.W. Vincent [a,*], J. Liu [a], M. Mohania [b]

[a] University of South Australia, Adelaide, Australia
[b] IBM Research Laboratory, New Delhi, India

ARTICLE INFO

Article history:
Received 11 January 2009
Received in revised form 15 November 2011
Accepted 9 January 2012
Available online 25 January 2012

Keywords:
XML
DTD
Functional dependency
Implication
Axiom system

ABSTRACT

With the growing use of XML as a format for the permanent storage of data, the study of functional dependencies in XML (XFDs) is of fundamental importance in a number of areas such as understanding how to effectively design XML databases without redundancy or update problems, and data integration. In this article we investigate a particular type of XFD, called a *weak 'closest node' XFD*, that has been shown to extend the classical notion of a functional dependency in relational databases. More specifically, we investigate the implication problem for weak 'closest node' XFDs in the context of XML documents with no missing information. The implication problem is the most important one in dependency theory, and is the problem of determining if a set of dependencies logically implies another dependency. Our first, and main, contribution is to provide an axiom system for XFD implication. We prove that our axiom system is both sound and complete, and we then use this result to develop a sound and complete quadratic time closure algorithm for XFD implication. Our second contribution is to investigate the implication problem for XFDs in the presence of a Document Type Definition (DTD). We show that for a class of DTDs called *structured* DTDs, the implication problem for a set of XFDs and a structured DTD can be converted to the implication problem for a set of XFDs alone, and so is axiomatizable and efficiently solvable by the first contribution. We do this by augmenting the original set of XFDs with additional XFDs generated from the structure of the DTD.

1. Introduction

Integrity constraints are one of the oldest and most important topics in database research, and they find application in a variety of areas such as database design, data translation, query optimization and data storage [1]. With the adoption of the eXtensible Markup Language (XML) [2] as the industry standard for data interchange over the internet, and its increasing usage as a format for the permanent storage of data in database systems [3], the study of integrity constraints in XML has increased in importance in recent years (a recent survey of the topic has been given by Fan [4]). It has also been argued that integrity constraints in XML might be of even greater importance than integrity constraints in relational database systems [4]. This is because of the adoption of XML as the standard for electronic data interchange, and the importance of integrity constraints in specifying the semantics of data – a topic that has been identified as crucial in effective data integration [5].

While a number of different types of XML integrity constraints have been studied [4], the growing use of XML as a permanent storage format for data has recently motivated the study of *functional dependencies in XML* (XFDs), because of their central role in designing XML documents that avoid redundancy and update problems. The formal study of XFDs and

* Corresponding author.
E-mail addresses: millist.vincent@unisa.edu.au (M.W. Vincent), jixue.liu@unisa.edu.au (J. Liu), mkmukesh@in.ibm.com (M. Mohania).

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2012.01.003

normalization in XML was initiated by Arenas and Libkin [6–8], who defined an XFD using the notion of a *'tree tuple'*, which in turn is an extension of the total unnesting operator used in nested relations [9]. An alternative approach to defining an XFD, based on the concept of a *'closest node'*, has been introduced by us in recent work [10]. Our approach has similarities with the one adopted by Buneman et al. in their study of keys in XML [11], and we have shown that a 'closest node' XFD generalizes an absolute XML key as defined by Buneman et al. for the case of simple XML paths [10]. While a 'closest node' XFD has the same syntax as a 'tree tuple' XFD, i.e. a set of simple paths on the l.h.s. of the XFD and a single simple path on the r.h.s. of the XFD, the 'closest node' approach differs from the 'tree tuple' approach in how one chooses the nodes in the tree representation of an XML document to test for XFD satisfaction. In the 'closest node' approach, nodes must satisfy a spatial property called 'closest', which in turn has been shown to be an extension of the property that two data values appear in the same tuple of a relation [12], whereas in the 'tree tuple' approach the nodes must belong to a tuple generated from the total unnest of the XML tree.

More recently [12], we have extended the study of 'closest node' XFDs, but using slightly different semantics to that originally proposed in [10].[1] To distinguish between these two approaches, we refer to an XFD as defined in [10] as a *strong* 'closest node' XFD, and that used in [12] and in this article as a *weak* 'closest node' XFD. This terminology is deliberate, since it will be shown later that if an XML document satisfies a 'closest node' XFD using strong semantics, then it also satisfies the XFD using weak semantics but the converse is not true. The motivation for defining a weak 'closest node' XFD was to preserve the semantics of an FD when a complete relation is mapped to an XML document, and we showed that if a complete relation is first mapped to a nested relation by an arbitrary sequence of nest operations [9], and then directly to an XML document, the resulting XML document satisfies a weak 'closest node' XFD if and only if the relation satisfies the corresponding FD [12]. In contrast, the motivation for the strong 'closest node' XFD definition was to extend the semantics of an XML key as proposed by Buneman et al. [11], in a similar fashion to the way that an FD extends the notion of a key in relational databases [1].

In general, it is not possible to compare a 'tree tuple' XFD to either a strong, or weak, 'closest node' XFD. This is because a *Document Type Definition* (DTD) [2] is mandatory in the 'tree tuple' approach, but optional in the 'closest node' approach, and because the two approaches use different semantics for missing information. However, the two approaches are comparable when a DTD is present and the XML document contains no missing information with respect to the paths of the DTD. In this case, the definition of a 'tree tuple' XFD has been shown to be equivalent to that of a weak 'closest node' XFD (but not in general to a strong 'closest node' XFD) [12].

In this article we focus on the *implication problem* for weak 'closest node' XFDs, that is determining if a set of XFDs logically implies another XFD. The implication problem is probably the most important one in dependency theory, irrespective of the data model or type of integrity constraint, since a solution to it lies at the heart of any automated procedure for reasoning about integrity constraints. In the context of 'tree tuple' XFDs, the implication problem has been investigated by Arenas and Libkin [6], and more recently by Kot and White [13]. Arenas and Libkin showed that the implication problem was efficiently decidable for restricted classes of DTDs, but was intractable for more general classes of DTDs. Kot and White developed a variant of the classical relational chase algorithm [1] to solve the XFD implication problem, both with and without a DTD. They then used their chase algorithm to develop sound and complete axiom systems for several classes of DTDs.

1.1. Approach

In this section we outline the general features of our approach. First, our definition of a weak 'closest node' XFD is the same as that given by us in related work [12]. In this approach a DTD is optional, and we investigate the implication problem for XFDs both in XML documents without a DTD, and in the presence of a new type of DTD that we call a *structured* DTD, which is a special case of a *simple* DTD as defined by Arenas and Libkin [6]. Our approach is similar to that of Kot and White [13] just mentioned in that they also consider XFD implication both in the presence and absence of a DTD, but differs from the approach of Arenas and Libkin who only consider XFD implication in the presence of a DTD. However, we only consider structured DTDs, rather than the more general classes of DTDs considered by both Arenas and Libkin and Kot and White, since it is the most appropriate class for the applications of XML that are the focus of this article, namely *data-centric* applications[14–17]. We shall discuss this point in more detail in Section 6.

Another feature of our approach is that we consider the XFD implication problem in the context of a new subclass of XML documents that we call *complete* XML documents,[2] which differs from the approach of Arenas and Libkin, and Kot and White, who allow incomplete XML documents. Intuitively, a complete XML document is one with 'no missing information' and is an extension of the notion of a complete relation. We now outline our motivation for investigating the implication problem in this context.

While one of the goals in the design of XML was to explicitly cater for irregularly structured data, XML is also being widely used in more traditional business applications involving regularly structured data, often referred to as data-centric XML [14–17]. For example, a recent survey of several hundred large companies in the U.S. found that around 70% were now using XML enabled databases, or native XML databases, for their core data processing applications [18]. In this setting,

[1] A detailed discussion of the differences between the two definitions will be given in a later section.

[2] A precise definition will be given later.

complete XML documents are a natural and important subclass, just as complete relations are in relational databases. We also note that a complete XML document differs from a complete relation in that it is not an absolute notion, rather it is only defined w.r.t. a specific set of paths which we assume to be given. However, similar to the relational case, our definition of logical implication is independent of the specific set of paths with respect to whom an XML document is complete.

The advantage of considering XFDs and their implication problem in complete XML documents, rather than arbitrary XML documents as considered by Kot and White [6,13], is similar to what occurs for FDs in relational databases [9,19]. In the relational setting, if a relation is complete then one obtains the well-known Armstrong axiom system for FD implication [1], whereas if a relation is incomplete then one obtains a weaker axiom system in which the transitivity rule in Armstrong's system is replaced by the weaker pseudo-transitivity rule [9,19]. Similarly, by restricting attention to complete XML documents, we will show that our axiom system is more powerful than the one obtained by Kot and White for arbitrary XML documents. By this we mean that in the context of complete XML documents, the axioms of Kot and White can be derived from our axioms, but the converse is not true since some of our axioms are not sound in arbitrary XML documents. Hence the axiom system of Kot and White is still sound, but no longer complete, for complete XML documents.

We observe that although the motivation for our approach is to extend the notions of FDs and FD implication in complete relations to the XML setting, our context and results are strict extensions and not simply translations of the definitions and results from the relational setting to XML. We now discuss this point in more detail. Firstly, our notion of a complete XML document is more general than that of a complete relation. This is because a complete XML document can contain duplicate information and also, when the document is modeled as a tree structure, the tree can contain leaf nodes which are not text values. Such documents cannot be generated from a complete relation. Secondly, although we only consider the implication problem for XFDs in complete XML documents, our definition of an XFD applies to any XML document. However, we note that although our definition of an XFD coincides with that of a 'tree tuple' XFD in complete XML documents [12], it has different semantics in arbitrary XML documents. This situation is similar to what occurs in relational databases, where there are several definitions of FD satisfaction in incomplete relations [9,19]. Thirdly, our axiom system for XFD implication differs substantially from Armstrong's axioms [1], and contains 5 additional rules that have no counterpart in the relational setting.

The final feature of our approach is that although the presence of a DTD is optional, we do require the existence of a set of paths that are legal for an XML document in the case where there is no DTD. This set could be generated from prior schema information, or it could be derived from the XML document itself if no such prior information exists. We also note that our approach requires that the set of paths be finite, so this excludes recursive DTDs in the case where the set of paths is derived from a DTD.

1.2. Contributions

The main contribution of this article is to solve the implication problem for weak 'closest node' XFDs in complete XML documents. In more detail, we make the following contributions.

- When the set of XFDs has the property that no path on the l.h.s. of an XFD ends in an element label (what we call a set of *text* XFDs), we show that the implication problem for 'closest node' XFDs has a sound and complete axiom system, and is efficiently solvable in quadratic time using a closure style algorithm.
- We then show that the implication problem for an arbitrary set of XFDs can be converted to the implication problem for a set of text XFDs. As a result, the implication problem for sets of arbitrary XFDs has a sound and complete axiom system and is solvable in quadratic time.
- We investigate the implication problem for 'closest node' XFDs in the presence of a structured DTD, and we show that this implication problem can be converted to the implication problem for a set of XFDs alone by adding XFDs derived from the structure of the DTD to the original set of XFDs. So, using the results just mentioned, this implication problem has a sound and complete axiom system and is efficiently solvable.
- Our completeness proofs use a chase style procedure on XML trees, and we also show that the procedure is an alternative method to our closure algorithm for computing XFD implication.
- In establishing our results, we introduce some novel techniques that might have application to the study of other integrity constraints in XML. In particular, showing that the implication problem for arbitrary XFDs can be reduced to the implication for text XFDs is a technique which, to the best of our knowledge, has not been used before.

The rest of this article is organized as follows. In Section 2 we present some preliminary definitions and the definition of a weak 'closest node' XFD. In Section 3 we present our axiom system for XFD implication and show that the system is sound. In order to establish completeness of the axiom system, the concept of a text XFD is presented in Section 4.1 and we show that the implication problem for a set of arbitrary XFDs can be converted to the implication problem for a set of text XFDs. In Section 4.2 we establish the main result of the article, which shows that our axiom system is both sound and complete, and that the XFD implication problem is solvable in quadratic time. The implication problem for XFDs in the presence of a DTD is investigated in Section 5, and it is shown that for the class of structured DTDs, this implication problem can be converted to the implication problem for an augmented set of XFDs without a DTD. In Section 6 we survey related work, and in Section 7 we discuss extensions to our work. Finally, Section 8 contains concluding comments.

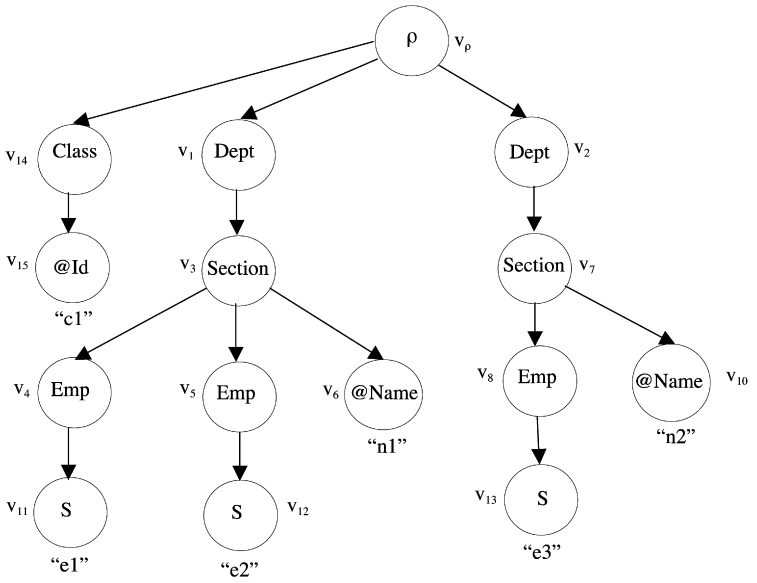

Fig. 1. An XML tree.

2. Basic concepts

In the section we present some preliminary definitions and then define a weak 'closest node' XFD.
We model an XML document as a tree, based on the model used in DOM and XPath [20,21], as follows.

Definition 1. Assume a countably infinite set **E** of element labels (tags), a countably infinite set **A** of attribute names such that **E** and **A** are mutually disjoint, and a symbol S indicating text such that $\mathtt{S} \notin \mathbf{E} \cup \mathbf{A}$. An *XML tree* $\mathbb{T}$ is defined by $\mathbb{T} \stackrel{\text{def}}{=} (\mathbf{E}, \mathbf{A}, V, lab, ele, att, val, \rho, v_\rho)$ where:

(i) V is a *finite* and non-empty ordered set of *nodes*;
(ii) lab is a total function $lab : V \mapsto \mathbf{E} \cup \mathbf{A} \cup \{\mathtt{S}\}$ with the property that $\forall v \in V$, if $lab(v) \in \mathbf{A} \cup \{\mathtt{S}\}$ then v is a leaf node in V;
(iii) ele is a partial function from V to a sequence of nodes in V such that $\forall v \in V$, if $v' \in ele(v)$ then $lab(v') \in \mathbf{E} \cup \{\mathtt{S}\}$;
(iv) att is a partial function $att : V \times \mathbf{A} \mapsto V$ such that $\forall v \in V$ and $a \in \mathbf{A}$, if $att(v, a) = v'$ then $lab(v) \in \mathbf{E}$ and $lab(v') = a$;
(v) $\rho \in \mathbf{E}$, and v_ρ is a distinguished node in V, called the *root node* of V, such that v_ρ is the only node in $\mathbb{T}$ where $lab(v_\rho) = \rho$;
(vi) val is a function such that $\forall v \in V, val(v) \stackrel{\text{def}}{=} v$ if $lab(v) \in \mathbf{E}$ and $val(v)$ is a string if either $lab(v) = \mathtt{S}$ or $lab(v) \in \mathbf{A}$;
(vii) the parent-child edge relation on V, $\{(v_1, v_2) | v_2$ occurs in $ele(v_1)$ or $v_2 = att(v_1, a)$ for some $a \in \mathbf{A}\}$ is required to form a tree rooted at v_ρ.

We also need the following tree-related notions.

Definition 2. Given $\mathbb{T} \stackrel{\text{def}}{=} (\mathbf{E}, \mathbf{A}, V, lab, ele, att, val, \rho, v_\rho)$:

- the parent node of a node $v \in V$ is denoted by $parent(v)$;
- the set of ancestors of a node $v \in V$ is denoted by $ancestor(v)$;
- $aancestor(v)$ is the set of nodes defined by $aancestor(v) = ancestor(v) \cup \{v\}$.

An XML tree is shown in Fig. 1, where $\mathbf{E} = \{\rho, \mathtt{Dept}, \mathtt{Section}, \mathtt{Emp}, \mathtt{Class}\}$ and $\mathbf{A} = \{\mathtt{@Name}, \mathtt{@Id}\}$.

Our model is the same as the one used by us in previous work [10,12], but differs slightly from the model used in defining a 'tree tuple' XFD [6,13]. The 'tree tuple' model assumes that an XML tree contains only unmixed data, that is an element node can have either element nodes or text nodes as children, but not both, whereas our model allows an element node to have both element nodes and text nodes as children.

There are a couple of other features of our model which we now briefly discuss. In order to model value equality semantics for text and attribute nodes, and node equality semantics for element nodes, the function *val* returns the id of a node for an element node, but the text value for a text or attribute node. This is similar to the approach used in the model

for 'tree tuple' XFDs [6,13]. Another feature of our approach is that although sibling nodes are ordered in our model, the definition of an XFD is independent of the ordering. Finally, in contrast to the terminology used by XPath [21], an attribute node in our model is defined to be a child of its parent element node.

We now give some preliminary definitions related to paths.

Definition 3. Given an XML tree $\mathbb{T} = (\mathbf{E}, \mathbf{A}, V, lab, ele, att, val, \rho, v_\rho)$, a *label-sequence* l is a sequence of labels that is either empty, denoted by $l = \epsilon$, or $l \stackrel{\text{def}}{=} l_1.\cdots.l_n$, where $n \geqslant 1$ and $l_i \in \mathbf{E} \cup \mathbf{A} \cup \{\mathtt{S}\}$ for every $i \in [1, n]$.[3]

Given label-sequences $l = l_1.\cdots.l_n$ and $l' = l'_1.\cdots.l'_m$, the *concatenation* of l and l', denoted by $l * l'$, is defined by $l * l' \stackrel{\text{def}}{=} l_1.\cdots.l_n.l'_1.\cdots.l'_m$.

Note that the concatenation of a label-sequence l and the empty sequence ϵ has the property that $l * \epsilon = \epsilon * l = l$.

Using the example in Fig. 1, $l_1 = \rho\,.\,\mathtt{Class}\,.\,\mathtt{@Id}$ and $l_2 = \mathtt{Emp}\,.\,\mathtt{S}$ are both label-sequences and $l_1 * l_2 = \rho\,.\,\mathtt{Class}\,.\,\mathtt{@Id}\,.\,\mathtt{Emp}\,.\,\mathtt{S}$.

Definition 4. A *path* is a *non-empty* label-sequence $l_1.\cdots.l_n$ such that $l_1 = \rho$ and $l_i \in (\mathbf{E} - \{\rho\})$ for all $i \in [2, n-1]$.

- Given a path $p = l_1.\cdots.l_n$, the function $Last(p)$ is defined by $Last(p) \stackrel{\text{def}}{=} l_n$; and the function $Parent(p)$ is defined by $Parent(p) \stackrel{\text{def}}{=} l_1.\cdots.l_{n-1}$ when $n > 1$, and undefined otherwise.
- Given paths $p = l_1.\cdots.l_n$ and $p' = l'_1.\cdots.l'_m$, p' is said to be a *prefix* of p, denoted by $p' \succeq p$, if there exists a label-sequence p'' such that $p' * p'' = p$.
- The path p' is said to be a *strict prefix* of p, denoted by $p' \succ p$, if $p' \succeq p$ and $p \neq p'$.
- The *intersection* of two paths p and p', denoted by $p \cap p'$, is the longest common prefix of both paths, which is also a path since ρ is a prefix of every path.

For example, the label-sequence l_1 above is a path but not l_2 since it does not start with the label ρ. Also, $Last(l_1) = \mathtt{@Id}$ and $Parent(l_1) = \rho\,.\,\mathtt{Class}$. Given paths $p_1 = \rho\,.\,\mathtt{Dept}\,.\,\mathtt{Section}\,.\,\mathtt{Emp}$ and $p_2 = \rho\,.\,\mathtt{Dept}\,.\,\mathtt{Section}\,.\,\mathtt{@Name}$, $p_1 \cap p_2 = \rho\,.\,\mathtt{Dept}\,.\,\mathtt{Section}$.

Definition 5. A *path instance* $\tilde{v}$ in an XML tree $\mathbb{T} = (\mathbf{E}, \mathbf{A}, V, lab, ele, att, val, \rho, v_\rho)$ is a *non-empty* sequence of nodes in V defined by $\tilde{v} \stackrel{\text{def}}{=} v_1.\cdots.v_n$, $n \geqslant 1$, such that $v_1 = v_\rho$ and for all $j \in [2, n]$, $v_j \in V$ and $parent(v_j) = v_{j-1}$.

- Given a path instance $\tilde{v} = v_1.\cdots.v_n$, the function $last(\tilde{v})$ is defined by $last(\tilde{v}) \stackrel{\text{def}}{=} v_n$; and the function $parent(\tilde{v})$ is defined by $parent(\tilde{v}) \stackrel{\text{def}}{=} v_1.\cdots.v_{n-1}$ if $n > 1$, and undefined otherwise.
- A path instance $\tilde{v} = v_1.\cdots.v_n$ is said to be *defined over the path* $p = l_1.\cdots.l_n$ if for all $j \in [1, n]$, $lab(v_j) = l_j$.
- The path instance $\tilde{v}' = v'_1.\cdots.v'_m$ is said to be a *prefix* of the path instance $\tilde{v} = v_1.\cdots.v_n$, denoted by $\tilde{v}' \succeq \tilde{v}$, if $m \leqslant n$ and $v_j = v'_j$ for all v'_j in $\tilde{v}'$.
- The set of path instances defined over a path p in $\mathbb{T}$ is denoted by $Inst(p, \mathbb{T})$.
- The *intersection* of two path instances $\tilde{v}$ and $\tilde{v}'$, denoted by $\tilde{v} \cap \tilde{v}'$, is the longest common prefix of $\tilde{v}$ and $\tilde{v}'$. This is also a path instance since the node v_ρ is a prefix of every path instance.

Using the example in Fig. 1, $\tilde{v} = v_\rho.v_1.v_3$ is a path instance, and $last(\tilde{v}) = v_3$ and $parent(\tilde{v}) = v_\rho.v_1$. Also, $Inst(\rho\,.\,\mathtt{Dept}, \mathbb{T}) = \{v_\rho.v_1, \rho.v_2\}$ and if $\tilde{v}_1 = v_\rho.v_1.v_3.v_4$ and $\tilde{v}_2 = v_\rho.v_1.v_3.v_5.v_{12}$, then $\tilde{v}_1 \cap \tilde{v}_2 = v_\rho.v_1.v_3$.

Our approach to defining an XFD does not require the existence of a DTD, but does require the existence of a *finite* and non-empty set of paths $\mathbb{P}$ that is legal for the XML document. The set $\mathbb{P}$ might be obtained from prior information, such as the relational schema from which the XML document has been mapped if such a schema exists, or simply from the XML document itself if no such prior information exists. We also place the following restriction on $\mathbb{P}$.[4]

Definition 6. Given paths p and p', a set of paths $\mathbb{P}$ is *downward closed* if whenever there exist paths p and p' such that $p \in \mathbb{P}$ and $p' \succ p$, then $p' \in \mathbb{P}$.[5]

We now define the notion of an XML tree conforming to $\mathbb{P}$, which is a weaker notion than that of an XML tree conforming to a DTD. By this we mean that if an XML tree conforms to a DTD then it also conforms to the set of paths of the DTD, but an XML tree can conform to the set of paths of a DTD without it conforming to the DTD.

[3] $[1, n]$ denotes the set $\{1, \ldots, n\}$.

[4] The downward closed property is called prefix-closed by Kot and White [13].

[5] For the rest of this article, unless stated otherwise we will assume that the set of paths $\mathbb{P}$ is downward closed.

Definition 7. Let $\mathbb{P}$ be a set of paths and let $\mathbb{T}$ be an XML tree. Then $\mathbb{T}$ is said to *conform* to $\mathbb{P}$, denoted by $\mathbb{T} \models \mathbb{P}$, if every path instance in $\mathbb{T}$ is a path instance defined over a path in $\mathbb{P}$.

We note that since the set of paths $\mathbb{P}$ is finite, and a path is a finite sequence of labels, then the set of labels appearing in any XML tree that conforms to $\mathbb{P}$ must be finite. Also, the restriction that $\mathbb{P}$ is finite excludes the case where the set of paths is generated from a recursive DTD.

As mentioned in Section 1, in this article we consider only XML trees with no missing information, which are defined as follows.

Definition 8. Let $\mathbb{P}$ be a set of paths, and let $\mathbb{T}$ be an XML tree such that $\mathbb{T} \models \mathbb{P}$.

- The function $[\![p]\!]$, where $p \in \mathbb{P}$, is the set of nodes defined by $[\![p]\!] \stackrel{\text{def}}{=} \{last(\tilde{v}) \mid \tilde{v} \in Inst(p, \mathbb{T})\}$.
- $\mathbb{T}$ is *complete* w.r.t. $\mathbb{P}$ if whenever there exist paths p' and p in $\mathbb{P}$ such that $p' \succ p$ and there exists a node $v' \in [\![p']\!]$, then there exists a node $v \in [\![p]\!]$ such that $v' \in ancestor(v)$.

For example, in Fig. 1, $[\![\rho\,.\,\texttt{Dept}]\!] = \{v_1, v_2\}$ and $[\![\rho\,.\,\texttt{Dept.Section.Emp.S}]\!] = \{v_{11}, v_{12}, v_{13}\}$. Also, if $\mathbb{P} = \{\rho,$ $\rho\,.\,\texttt{Dept}, \rho\,.\,\texttt{Dept.Section}, \rho\,.\,\texttt{Dept.Section.Emp}, \rho\,.\,\texttt{Class}, \rho\,.\,\texttt{Dept.Section.Emp.S}, \rho\,.\,\texttt{Dept.Section.@Name}, \rho\,.\,\texttt{Class.@Id}\}$ then the tree in Fig. 1 conforms to $\mathbb{P}$ and is complete w.r.t. $\mathbb{P}$. We also note that if, for example, node v_{10} is removed, then the new tree will conform to $\mathbb{P}$ but will not be complete w.r.t. $\mathbb{P}$.

We emphasize that the notion of an XML tree being complete is not an absolute notion, it is only defined relative to a specific set of paths. Thus it is possible that a tree $\mathbb{T}$ conforms to a set of paths $\mathbb{P}$ and is complete w.r.t. $\mathbb{P}$, but $\mathbb{T}$ is not complete to another set of paths $\mathbb{P}'$ even though $\mathbb{T}$ also conforms to $\mathbb{P}'$.

We note that if $\mathbb{T}$ is a complete tree such that $\mathbb{T} \models \mathbb{P}$, then for every path $p \in \mathbb{P}$, $[\![p]\!] \neq \emptyset$, since $\mathbb{T}$ always contains the root node v_ρ and every path starts with the label ρ. We also note that for any $\mathbb{P}$ there always exists a complete tree $\mathbb{T}$ such that $\mathbb{T} \models \mathbb{P}$ [12]. Unless stated otherwise, for the rest of the article we assume that all XML trees are complete. We now present additional definitions that are needed to define a 'closest node' XFD.

The next definition is central to the definition of a 'closest node' XFD.

Definition 9. Let $\mathbb{P}$ be a set of paths, and let $\mathbb{T}$ be an XML tree such that $\mathbb{T} \models \mathbb{P}$.

If v_i and v_j are two nodes (not necessarily distinct) in $\mathbb{T}$, the boolean function $closest(v_i, v_j)$ returns true if there exists a node x_j^i in $\mathbb{T}$ such that $x_j^i \in aancestor(v_i)$ and $x_j^i \in aancestor(v_j)$ and $x_j^i \in [\![p_i \cap p_j]\!]$ (there can be at most one such node x_j^i since $\mathbb{T}$ is a tree), where p_i is the path in $\mathbb{P}$ such that $v_i \in [\![p_i]\!]$ and p_j is the path in $\mathbb{P}$ such that $v_j \in [\![p_j]\!]$.

Essentially, two nodes v_i and v_j are 'closest' if the two nodes have a common *aancestor* node in $[\![p_i \cap p_j]\!]$, where $v_i \in [\![p_i]\!]$ and $v_j \in [\![p_j]\!]$. For example $closest(v_5, v_6)$ is true in Fig. 1 since if we let $x_6^5 = v_3$, then $v_3 \in aancestor(v_5)$, $v_3 \in aancestor(v_6)$, $v_3 \in [\![\rho\,.\,\texttt{Dept.Section}]\!]$, $\rho\,.\,\texttt{Dept.Section} = \rho\,.\,\texttt{Dept.Section.Emp} \cap \rho\,.\,\texttt{Dept.Section.@Name}$, $v_5 \in [\![\rho\,.\,\texttt{Dept.Section.Emp}]\!]$ and $v_6 \in [\![\rho\,.\,\texttt{Dept.Section.@Name}]\!]$. Also, $closest(v_2, v_7)$ is true since if we let $x_7^2 = v_2$, then $v_2 \in [\![\rho\,.\,\texttt{Dept}]\!]$, $v_7 \in [\![\rho\,.\,\texttt{Dept.Section}]\!]$, $\rho\,.\,\texttt{Dept} \cap \rho\,.\,\texttt{Dept.Section} = \rho\,.\,\texttt{Dept}$ and v_2 is an *aancestor* of both v_2 and v_7; and $closest(v_4, v_4)$ is true since if we let $x_4^4 = v_4$, then $v_4 \in [\![\rho\,.\,\texttt{Dept.Section.Emp}]\!]$, $\rho\,.\,\texttt{Dept.Section.Emp} \cap \rho\,.\,\texttt{Dept.Section.Emp} = \rho\,.\,\texttt{Dept.Section.Emp}$ and v_4 is an *aancestor* of v_4.

However, $closest(v_5, v_{10})$ is false since the only node that is an *aancestor* of both v_5 and v_{10} is v_ρ, but $v_\rho \in [\![\rho]\!]$ and $\rho\,.\,\texttt{Dept.Section.Emp} \cap \rho\,.\,\texttt{Dept.Section.@Name} = \rho\,.\,\texttt{Dept.Section} \neq \rho$; and $closest(v_4, v_5)$ is false since the only nodes which are an *aancestor* of both v_4 and v_5 are v_3, v_1, ρ, but none of these nodes belong to $\rho\,.\,\texttt{Dept.Section.Emp}$ which is the intersection of $\rho\,.\,\texttt{Dept.Section.Emp}$ with itself.

We observe that for any node v in an XML tree, $closest(v, v)$ is true. Also, for any nodes v and v' in an XML tree, $closest(v, v')$ is equivalent to $closest(v', v)$, and if v is either an ancestor or descendant of v' then $closest(v, v')$ is true. However the converse is not true, i.e. $closest(v, v')$ can be true without v and v' having either a descendant or ancestor relationship.

Importantly, we also observe that *closest* is not transitive and so $closest(v, v')$ and $closest(v', v'')$ does not necessarily imply $closest(v, v'')$. For example, in Fig. 1 $closest(v_6, v_{14})$ and $closest(v_{14}, v_8)$ are true but not $closest(v_6, v_8)$.

The *closest* property is related to the concept of *locality*, an important concept used in finite model theory for determining the expressive power of query languages [22]. In the relational model, two data values in a relation are said to be local if there is a tuple to which they both belong. We have recently shown that the *closest* property is equivalent to the locality property when a complete relation is first mapped to a nested relation by an arbitrary sequence of nest operations, and then directly to an XML tree [12]. In other words, two data values in a relation are local if and only if the corresponding nodes in the XML document satisfy the *closest* property.

We now extend the notion of two nodes being closest to a set of nodes pairwise satisfying the closest property, called a *cn-set*.

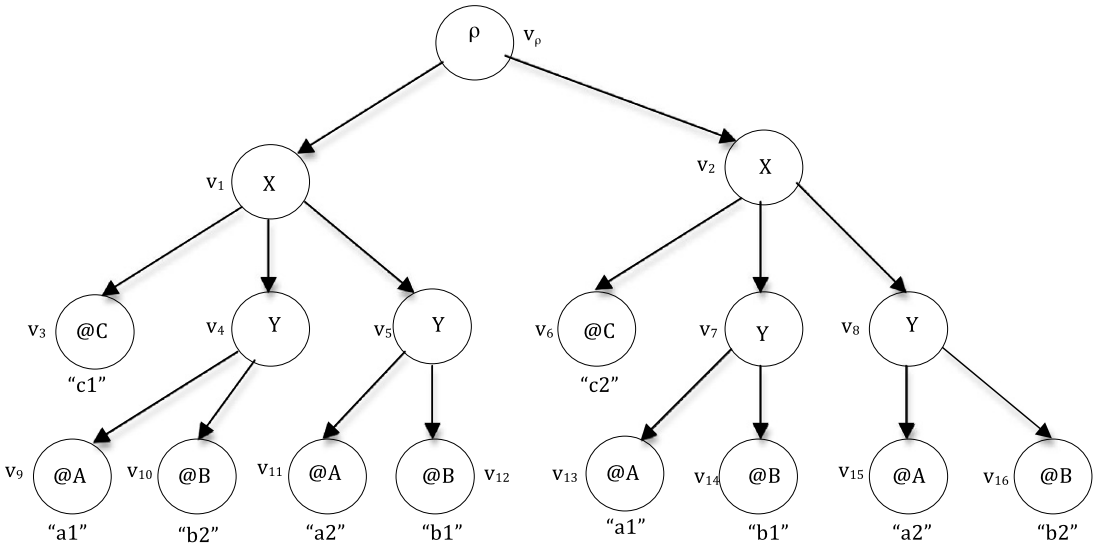

Fig. 2. An XML tree.

Definition 10. Let $\mathbb{P}$ be a set of paths, and let $\mathbb{T}$ be an XML tree such that $\mathbb{T} \models \mathbb{P}$. A set of nodes $\{v_1, \ldots, v_n\}$ in $\mathbb{T}$ is defined to be a cn-set if for all $i, j \in [1, n]$, $closest(v_i, v_j)$ is true.

For example in Fig. 1, v_{15}, v_1, v_3[6] is a cn-set, but v_{15}, v_1, v_2 is not since $closest(v_1, v_2)$ is not true. We also note that any subset of a cn-set is also a cn-set. We now define a weak 'closest node' XFD, following our previous work [12].

Definition 11. Let $\mathbb{P}$ be a set of paths and let $\mathbb{T}$ be an XML tree (not necessarily complete) such that $\mathbb{T} \models \mathbb{P}$.

A weak 'closest node' XFD is a statement of the form $\{p_1, \ldots, p_n\} \rightarrow \{p_{n+1}, \ldots, p_{n+m}\}$, where $\{p_1, \ldots, p_{n+m}\} \subseteq \mathbb{P}$, and $n \geqslant 1$ and $m \geqslant 1$.

The tree $\mathbb{T}$ satisfies the XFD if whenever there exist nodes $\{v_1, v'_1\} \subseteq [\![p_1]\!], \ldots, \{v_{n+m}, v'_{n+m}\} \subseteq [\![p_{n+m}]\!]$ in $\mathbb{T}$ such that the following hold:

(i) $v_1, \ldots, v_{n+m}$ and $v'_1, \ldots, v'_{n+m}$ are cn-sets;
(ii) for all $i \in [1, n]$, $val(v_i) = val(v'_i)$

then for all $i \in [n+1, n+m]$, $val(v_i) = val(v'_i)$.

We note that because both the l.h.s. and r.h.s. of an XFD are sets of paths, duplicate paths are not allowed in an XFD and the paths are not ordered. Also, to simplify the notation, for the rest of the article we will write the XFD $\{p_1, \ldots, p_n\} \rightarrow \{p_{n+1}, \ldots, p_{n+m}\}$ simply as $p_1, \ldots, p_n \rightarrow p_{n+1}, \ldots, p_{n+m}$.

To illustrate the definition, we claim that the XFD ρ.Dept.Section.@Name $\rightarrow \rho$.Dept.Section.Emp.S is violated in Fig. 1. This is because if we take the pairs of nodes v_6, v_{11} and v_6, v_{12}, then $closest(v_6, v_{11})$ is true, $closest(v_6, v_{12})$ is true, and $val(v_{11}) =$ "$e1$" $\neq$ "$e2$" $= val(v_{12})$. We note that if $val(v_{12})$ was changed to "$e1$", then the XFD would be satisfied.

There are several features of our XFD definition that we now discuss. Firstly, as mentioned in Section 1, our definition differs from the definition of a 'closest node' XFD as originally presented by us [10], which we refer to as a strong 'closest node' XFD, and is a weaker notion. This is because in the definition of a strong 'closest node' XFD, the only pairs of nodes required to satisfy the *closest* property are pairs consisting of any node from the l.h.s. of the XFD and the node on the r.h.s. of the XFD (assuming for the moment that there is only one path on the r.h.s. of the XFD). However, in a weak 'closest node' XFD, all pairs of nodes are required to satisfy the *closest* property, including nodes corresponding to paths that are both on the l.h.s. of the XFD. Consequently, any XML tree that satisfies a strong 'closest node' XFD also satisfies the weak 'closest node' XFD, but the reverse does not necessarily hold. We illustrate this point in the following example.

Example 12. Consider the XFD ρ.X.Y.@A, ρ.X.Y.@B $\rightarrow \rho$.X.@C in Fig. 2. Using the strong 'closest node' semantics given in our previous work [10], the XFD is violated because in considering the nodes v_9, v_{12}, v_3 and v_{13}, v_{14}, v_6; $closest(v_9, v_3)$ is true, $closest(v_{12}, v_3)$ is true, $closest(v_{13}, v_6)$ is true, $closest(v_{14}, v_6)$ is true, v_9 and v_{13} have the same *val*, v_{12} and v_{14}

[6] To simplify the presentation, for the rest of this article we shall not use the normal set delimiters { and } when presenting the nodes in a cn-set.

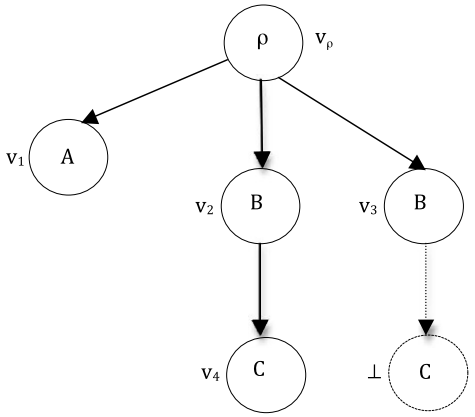

Fig. 3. An XML tree.

have the same *val*, but v_3 and v_6 have different *val*'s. However, the XFD is satisfied using the weak 'closest node' semantics of this paper because the set of nodes v_9, v_{12}, v_3 is not a cn-set since $closest(v_9, v_{12})$ does not hold.

Next, the motivation for the definition of a strong 'closest node' XFD was to extend the notion of an XML key as defined by Buneman et al. [11], and we showed that this property does indeed hold for the case of simple paths [10]. However, this extension property no longer holds for a weak 'closest node' XFD and an XML key as defined by Buneman et al., and instead a weak 'closest node' XFD extends the notion of a relational FD [12]. Thus, unlike the relational case where a key constraint is a special case of an FD, the notions of an XML key and a weak 'closest node' XFD, and thus also a 'tree tuple' XFD [12], are separate notions in XML.

Finally, we note that although the motivation for defining a weak 'closest node' XFD was to extend the notion of a relational FD, a weak 'closest node' XFD is more general than an FD in several aspects. Firstly, a weak 'closest node' XFD is defined for arbitrary XML documents, including those with missing and duplicate information as shown in Example 13 below, which is a larger class of XML documents than those obtained from complete relations. We also note that the class of complete XML documents is a larger class than those obtained from complete relations, since a complete XML document can contain duplicate data. Secondly, a weak 'closest node' XFD allows paths that end with element labels as well as text and attribute labels, whereas an XFD obtained from mapping a relation to XML contains only paths that end with an attribute or text label. So a weak 'closest node' XFD combines value and identity semantics, whereas an FD uses value semantics alone.

Finally, although we have shown elsewhere that a weak 'closest node' XFD and a 'tree tuple' XFD are equivalent for XML documents which are complete w.r.t. the set of paths [12], their semantics do not coincide in XML documents with missing information. The reason for this is that our definition of 'closest node' XFD satisfaction is an extension of the notion of 'weak satisfaction' of an FD in incomplete relations where a null is interpreted as an 'existing but unknown' value [9,19]. In this approach, an FD is defined to be satisfied as long as there is no 'hard' violation of the FD constraint, i.e. a violation not involving nulls. Similarly, we define a weak 'closest node' XFD to be satisfied as long as there is no 'hard' violation. Our approach is similar to that adopted by Buneman et al. in their work on XML keys [11], since they also define an XML key to be satisfied as long as there is no 'hard' violation.

In contrast, the semantics of a 'tree tuple' XFD is an extension of FD satisfaction in incomplete relations where a null is interpreted as a 'non-existent' or 'inapplicable' value [9]. In this approach, an FD $X \rightarrow A$ is violated when there exist two tuples which are non-null and equal on X, but one A value is null and the other is non-null. Similarly, a 'tree tuple' XFD is violated when there are two tree tuples which are non-null and equal on the paths on the left of the XFD, but one 'tree tuple' is null for the path on the right of the XFD but the other is non-null, whereas a 'closest node' XFD is satisfied in this case.

We now illustrate the difference between the two definitions by an example.

Example 13. Consider the XFD $\rho.\mathtt{A} \rightarrow \rho.\mathtt{B}.\mathtt{C}$ and Fig. 3 ($\perp$ represents a null node). According to weak 'closest node' semantics, the XFD is satisfied since there is only one cn-set, $v_1.v_4$, for the paths $\rho.\mathtt{A}, \rho.\mathtt{B}.\mathtt{C}$. However, according to 'tree tuple' semantics the XFD is violated since the node v_3 is considered to have a $\mathtt{B}$ child node with a null value, and so the two tree tuples t_1, t_2 defined by $\langle t_1(\rho.\mathtt{A}) = v_1, t_1(\rho.\mathtt{B}.\mathtt{C}) = v_4\rangle$ and $\langle t_2(\rho.\mathtt{A}) = v_1, t_2(\rho.\mathtt{B}.\mathtt{C}) = \perp\rangle$ agree on the path $\rho.\mathtt{A}$ but disagree on the path $\rho.\mathtt{B}.\mathtt{C}$.

This example also illustrates an important difference between the two approaches to defining XFD satisfaction, namely 'tree tuple' semantics requires the use of a special value to represent a null whereas 'closest node' semantics does not.

3. Axiom system for XFD implication

In this section we present an axiom system for reasoning about the implication of XFDs.[7] We first present some preliminary definitions.

Definition 14.[8]

- An XML tree $\mathbb{T}$ satisfies an XFD σ is denoted by $\mathbb{T} \models \sigma$, and $\mathbb{T}$ satisfies a set of XFDs Σ is denoted by $\mathbb{T} \models \Sigma$.
- An XFD $\sigma = p_1, \ldots, p_n \rightarrow p_{n+1}, \ldots, p_{n+m}$, is *consistent* with a set of paths $\mathbb{P}$, denoted by $\sigma \models \mathbb{P}$, if $\{p_1, \ldots, p_{n+m}\} \subseteq \mathbb{P}$. A set of XFDs Σ is consistent with $\mathbb{P}$, denoted by $\Sigma \models \mathbb{P}$, if $\forall \sigma \in \Sigma$, $\sigma \models \mathbb{P}$.
- Given a set of paths $\mathbb{P}$ and a set of XFDs Σ and an XFD σ such that $\Sigma \cup \{\sigma\} \models \mathbb{P}$, Σ *implies* σ, denoted by $(\mathbb{P}, \Sigma) \vdash \sigma$, if for every XML tree $\mathbb{T}$ such that $\mathbb{T} \models \mathbb{P}$ and $\mathbb{T} \models \Sigma$, then $\mathbb{T} \models \sigma$. We also define the set of XFDs implied by $(\mathbb{P}, \Sigma)$ by $(\mathbb{P}, \Sigma)^+ \stackrel{\text{def}}{=} \{\sigma \mid (\mathbb{P}, \Sigma) \vdash \sigma\}$.
- Two set of XFDs Σ and Σ' are equivalent, denoted by $(\mathbb{P}, \Sigma) \equiv (\mathbb{P}, \Sigma')$, if $(\mathbb{P}, \Sigma)^+ = (\mathbb{P}, \Sigma')^+$.
- If $\Sigma \models \mathbb{P}$ and $P \subseteq \mathbb{P}$, then the set of all paths implied by P is defined by $(\mathbb{P}, \Sigma, P)^+ \stackrel{\text{def}}{=} \{q \mid q \in \mathbb{P} \text{ and } P \rightarrow q \in (\mathbb{P}, \Sigma)^+\}$.
- The set of paths appearing on either the l.h.s. or the r.h.s. of any $\sigma \in \Sigma$ is denoted by $\mathbb{P}_\Sigma$.

We note that although our definition of XFD implication is w.r.t. a specific set of paths $\mathbb{P}$, we shall show later in this section that XFD implication is in fact independent of $\mathbb{P}$.

Our axiom system for XFD implication in complete XML trees is then the following (assuming that $\Sigma \models \mathbb{P}$).

Axiom A1 (Reflexive Rule): $(\mathbb{P}, \Sigma) \vdash p_1, \ldots, p_n \rightarrow p_i$, for all $i \in [1, n]$.
Axiom A2 (Augmentation Rule): If $(\mathbb{P}, \Sigma) \vdash p_1, \ldots, p_n \rightarrow q_1, \ldots, q_m$ then $(\mathbb{P}, \Sigma) \vdash p_0, p_1, \ldots, p_n \rightarrow p_0, q_1, \ldots, q_m$.
Axiom A3 (Transitivity Rule): If $(\mathbb{P}, \Sigma) \vdash p_1, \ldots, p_n \rightarrow q_1, \ldots, q_m$ and $(\mathbb{P}, \Sigma) \vdash q_1, \ldots, q_m \rightarrow s$ then $(\mathbb{P}, \Sigma) \vdash p_1, \ldots, p_n \rightarrow s$.
Axiom A4 (Intersection Rule): If $(\mathbb{P}, \Sigma) \vdash p_1, \ldots, p_n \rightarrow q$, and $p_i \cap q = \rho$ for all $i \in [1, n]$, then $(\mathbb{P}, \Sigma) \vdash p \rightarrow q$ for every path p.
Axiom A5 (Prefix Rule): If $(\mathbb{P}, \Sigma) \vdash p'_1, \ldots, p'_n \rightarrow q$ and there exist paths $p_1, \ldots, p_n$ (not necessarily distinct) such that for all $i \in [1, n]$:
(i) $p'_i \cap q \succeq p_i$; and
(ii) $p_i \succeq p'_i$ or $p_i \succeq q$
then $(\mathbb{P}, \Sigma) \vdash RED(p_1, \ldots, p_n) \rightarrow q$, where $RED(p_1, \ldots, p_n)$ denotes the set of paths obtained by removing duplicate paths from $p_1, \ldots, p_n$.
Axiom A6 (Ancestor Rule): If $Last(p) \in \mathbf{E}$ and $q \succ p$ then $(\mathbb{P}, \Sigma) \vdash p \rightarrow q$.
Axiom A7 (Attribute Rule): If $Last(p) \in \mathbf{A}$ then $(\mathbb{P}, \Sigma) \vdash Parent(p) \rightarrow p$.
Axiom A8 (Root Rule): $(\mathbb{P}, \Sigma) \vdash p_1, \ldots, p_n \rightarrow \rho$.

Axioms A1–A3 correspond to Armstrong's axioms for FDs [1], but the remaining axioms have no parallels in the axioms for FDs. We also note that although there is only a single path, s, on the r.h.s. of the final XFD in Axiom A3, Axioms A1–A3 can be combined to show that Axiom A3 still holds if the path s is replaced by a set of paths.

We now illustrate the use of Axioms A4 and A5.

Example 15. Consider the set of XFDs $\Sigma = \{\rho.\texttt{@Course} \rightarrow \rho.\texttt{Dept.Section.Emp.S}, \rho.\texttt{Dept.Section.Office.S} \rightarrow \rho.\texttt{Dept.Section.Emp.Name.@Fname}\}$ which is satisfied by the XML tree in Fig. 4. Considering then the XFD $\rho.\texttt{@Course} \rightarrow \rho.\texttt{Dept.Section.Emp.S}$, applying Axiom A4 shows that the XFDs $\rho \rightarrow \rho.\texttt{Dept.Section.Emp.S}$, $\rho.\texttt{Dept} \rightarrow \rho.\texttt{Dept.Section.Emp.S}$, etc., also hold. Also, from $\rho.\texttt{Dept.Section.Office.S} \rightarrow \rho.\texttt{Dept.Section.Emp.Name.@Fname}$ and applying Axiom A5 we deduce that the XFDs

$\rho.\texttt{Dept.Section.Office} \rightarrow \rho.\texttt{Dept.Section.Emp.Name.@Fname}$,
$\rho.\texttt{Dept.Section} \rightarrow \rho.\texttt{Dept.Section.Emp.Name.@Fname}$,
$\rho.\texttt{Dept.Section.Emp} \rightarrow \rho.\texttt{Dept.Section.Emp.Name.@Fname}$,
$\rho.\texttt{Dept.Section.Emp.Name} \rightarrow \rho.\texttt{Dept.Section.Emp.Name.@Fname}$
also hold.

We also note that both (i) and (ii) of Axiom A5 need to be satisfied for the axiom to hold. To illustrate this, considering $\rho.\texttt{Dept.Section.Office.S} \rightarrow \rho.\texttt{Dept.Section.Emp.Name.@Fname}$ then the XFD $\rho.\texttt{Dept.Section.Office.Emp.S} \rightarrow \rho.\texttt{Dept.Section.Emp.Name.@Fname}$ satisfies (i) of Axiom A5 but not (ii), however the XFD is violated in Fig. 4. Also, the XFD $\rho.\texttt{Dept} \rightarrow \rho.\texttt{Dept.Section.Emp.Name.@Fname}$ satisfies (ii) of Axiom A5 but not (i), yet it is also violated in Fig. 4.

[7] For the remainder of the article and unless stated otherwise, an XFD will denote a weak 'closest node' XFD, $\mathbb{P}$ will denote a downward closed set of paths, σ will denote a single weak 'closest node' XFD, Σ will denote a set of weak 'closest node' XFDs and $\mathbb{T}$ denote a complete XML tree w.r.t. $\mathbb{P}$.

[8] A summary of the notation used in this article is contained in Fig. A.9 at the end of Appendix A.

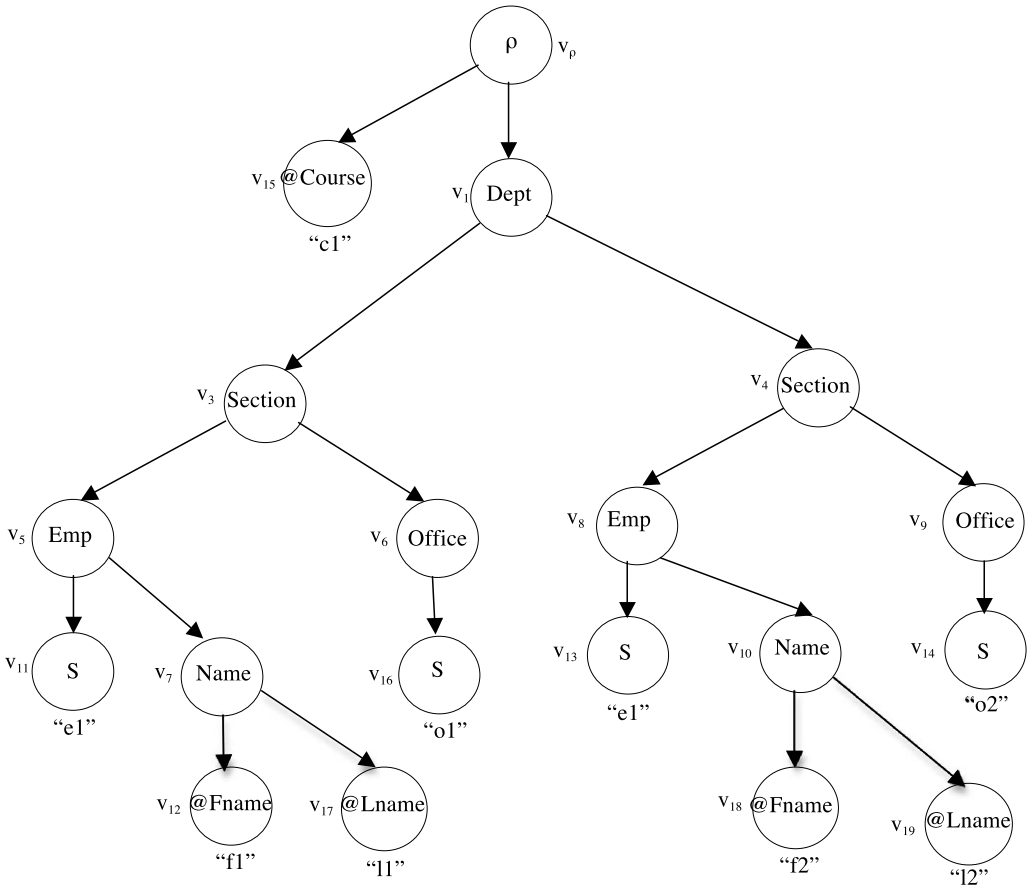

Fig. 4. An XML tree.

Also, although in this article we investigate the implication problem for XML trees which are complete w.r.t. a set of paths, we note that Axioms A4 and A5[9] are not sound in incomplete trees and now demonstrate this by an example.

Example 16. Consider the set of XFDs $\Sigma = \{\rho.\mathtt{X}.\mathtt{A} \to \rho.\mathtt{Y}.\mathtt{C}.\mathtt{S}, \rho.\mathtt{Y}.\mathtt{A}.\mathtt{B} \to \rho.\mathtt{Y}.\mathtt{C}.\mathtt{S}\}$ and Fig. 5. Then using weak 'closest node' semantics, both XFDs are satisfied since $[\![\rho.\mathtt{X}.\mathtt{A}]\!]$ and $[\![\rho.\mathtt{Y}.\mathtt{A}.\mathtt{B}]\!]$ are both empty in $\mathbb{T}$. However, from Axiom A4 and $\rho.\mathtt{X}.\mathtt{A} \to \rho.\mathtt{Y}.\mathtt{C}.\mathtt{S}$ we derive $\sigma_1 = \rho.\mathtt{X} \to \rho.\mathtt{Y}.\mathtt{C}.\mathtt{S}$, yet this XFD is violated because for the cn-sets v_1, v_6 and v_1, v_7; $val(v_6) =$ "c1" $\neq$ "c2" $= val(v_7)$. Also, from Axiom A5 and $\rho.\mathtt{Y}.\mathtt{A}.\mathtt{B} \to \rho.\mathtt{Y}.\mathtt{C}.\mathtt{S}$ we derive $\sigma_2 = \rho.\mathtt{Y}.\mathtt{A} \to \rho.\mathtt{Y}.\mathtt{C}.\mathtt{S}$, yet by considering the cn-sets v_3, v_6 and v_3, v_7 it follows that $\rho.\mathtt{Y}.\mathtt{A} \to \rho.\mathtt{Y}.\mathtt{C}.\mathtt{S}$ is violated.

The remaining XFD axioms are for *trivial* XFDs, that is axioms that hold in any complete XML tree. Axiom A6 specifies that a path ending in an element label determines any prefix path. This is a consequence of the fact that the *val* of an element node is its identifier, which is unique in the tree, and each node in an XML tree has only one ancestor node belonging to a specific prefix path. We note that Axiom A6 does not hold if the path p ends in a text or attribute label. For instance, the XFD $\rho.\mathtt{Dept}.\mathtt{Section}.\mathtt{Emp}.\mathtt{S} \to \rho.\mathtt{Dept}.\mathtt{Section}.\mathtt{Emp}$ does not hold in Fig. 4.

Axiom A7 specifies that the parent path of an attribute path determines the attribute path. This is a consequence of the fact that an element node can only have one attribute node for each element label. We also note that the parent path of an attribute path must always end in an element label, since only leaf nodes of an XML tree can be an attribute or text label. Finally, Axiom A8 specifies that any set of paths determines the root path. This is a consequence of the fact each tree has only one root node.

We also note that since XFDs obey Armstrong's axioms then, as for FDs [1], every XFD with multiple paths on the r.h.s. of the XFD is equivalent to a set of XFDs with the same l.h.s. and single paths on the r.h.s.

We now compare our axiom system with that obtained by Kot and White [13] for 'tree tuple' XFD implication in arbitrary XML trees and we show that in the case of complete XML trees, our axiom system is more powerful than that of Kot and White in the sense that our axiom system implies that of Kot and White, but not the reverse. To show the first part, we first

[9] Similar to the relational case, the transitivity Axiom A3 is also not sound in incomplete XML trees.

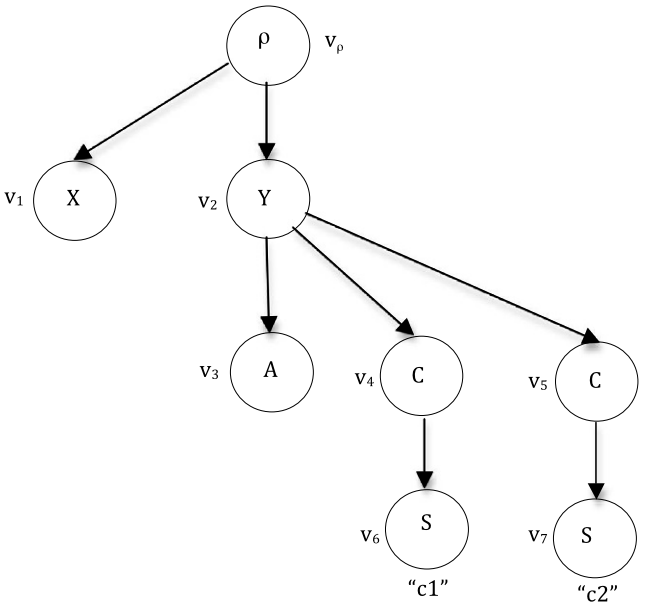

Fig. 5. An XML tree $\mathbb{T}$.

note that in the case of complete XML trees, axioms 1–13 of Kot and White reduce to axioms 4–7, and 10. Also, axioms 14–18 of Kot and White define the properties of an XML tree and follow from properties (i), (v), (iii) and (iv), (vii), and (iv) respectively of Definition 1. Next, axiom 4 follows from our Axiom A1, axiom 5 follows from our Axiom A2, axioms 6 and 7 follow from our Axioms A1–A3 (as for FDs), and axiom 10 follows from our Axiom A3. Next, axiom 15 follows from our Axiom A8, axiom 17 follows from our Axiom A6 and axiom 18 follows from our Axiom A7. Finally, axiom 19 of Kot and White can be derived from our Axiom A5 as follows. We first translate axiom 19 into our notation as follows:

19. If $(\mathbb{P}, \Sigma) \vdash p'_1, \ldots, p'_n, p'_{n+1}, \ldots, p'_{n+m} \rightarrow q$ then $(\mathbb{P}, \Sigma) \vdash q', p_{n+1}, \ldots, p_{n+k} \rightarrow q$, if:

(a) there is a label-sequence y such that $q' * y \succeq q$;
(b) $\forall p \in \{p'_1, \ldots, p'_n\}, q' \not\succeq p$;
(c) $\{p_{n+1}, \ldots, p_{n+k}\}$ is the subset of $\{p'_{n+1}, \ldots, p'_{n+m}\}$ such that for every path p in the subset, $q' * y \succeq p$

(re-labelling of subscripts might be necessary).

To show that axiom 19 follows from Axiom A5, we use another result, Lemma 19, which we will soon present. Consider first the XFD $\sigma = p_1, \ldots, p_n,\ p_{n+1}, \ldots, p_{n+k}, p_{n+k+1},\ \ldots, p_{n+m} \rightarrow q$, where for all $p \in \{p_1, \ldots, p_n,\ p_{n+k+1}, \ldots, p_{n+m}\}$, $p = q'$. If $i \in [1, n]$ or $i \in [n+k+1, n+m]$, then $p_i = q'$ and so (i) of Axiom A5 becomes $p'_i \cap q \succeq q'$. This condition holds since otherwise it would mean that $q' \succeq p'_i \cap q$ from Lemma 19 (vi) since $p'_i \cap q \succeq q$ from Lemma 19 (vii) and $q' \succeq q$ from (a), which implies that $q' \succeq p'_i$ from Lemma 19 (xi) because $p'_i \cap q \succeq p'_i$ from Lemma 19 (vii). This contradicts (b), and so (i) of Axiom A5 holds for this case, as also does (ii) since $p_i = q'$ and hence $q' \succeq q$ follows from (a). If instead $i \in [n+1, n+k]$, then $p_i = p'_i$ and so (i) of Axiom A5 becomes $p'_i \cap q \succeq p'_i$ which holds from Lemma 19 (vii), and (ii) holds since $p_i \succeq p'_i$ follows from $p_i = p'_i$. Hence Axiom A5 applies, and so we deduce that $(\mathbb{P}, \Sigma) \vdash RED(\sigma)$ which establishes axiom 19.

Next, our claim that the axiom system of Kot and White does not imply our axiom system is demonstrated by Example 16. In this example we have shown that our Axioms A4 and A5 are not sound, but since the tree $\mathbb{T}$ satisfies Σ and the axiom system of Kot and White is sound, then this means that Axioms A4 and A5 cannot be derived from the axiom system of Kot and White or else one would derive the contradiction that $\mathbb{T}$ satisfies σ_1 and σ_2.

Finally, we show that our definition of XFD implication is independent of the specific set of paths $\mathbb{P}$.

Lemma 17. *Let Σ be a set of XFDs, let σ be an XFD and let $\mathbb{P}$ and $\mathbb{P}'$ be sets of paths such that $\Sigma \cup \{\sigma\} \models \mathbb{P}$ and $\Sigma \cup \{\sigma\} \models \mathbb{P}'$. Then $(\mathbb{P}, \Sigma) \vdash \sigma$ iff $(\mathbb{P}', \Sigma) \vdash \sigma$.*

Proof. We shall first show that if $(\mathbb{P}, \Sigma) \nvdash \sigma$ then $(\mathbb{P}', \Sigma) \nvdash \sigma$. If $(\mathbb{P}, \Sigma) \nvdash \sigma$ then there exists a complete XML tree $\mathbb{T}$ such that $\mathbb{T} \models \mathbb{P}$, $\mathbb{T} \models \Sigma$ but $\mathbb{T} \not\models \sigma$. We shall establish the result by showing how to construct from $\mathbb{T}$ a complete XML tree $\mathbb{T}'$ such that $\mathbb{T}' \models \mathbb{P}'$, $\mathbb{T}' \models \Sigma$ but $\mathbb{T}' \not\models \sigma$. First, for every path $p \in \mathbb{P} - \mathbb{P}'$, remove all nodes in $[\![p]\!]$ from $\mathbb{T}$, resulting in an XML tree $\mathbb{T}_1$. Now since $\Sigma \cup \{\sigma\} \models \mathbb{P}$ and $\Sigma \cup \{\sigma\} \models \mathbb{P}'$ and $\mathbb{P}$ and $\mathbb{P}'$ are downward closed, p can neither be a path in $\Sigma \cup \{\sigma\}$ nor a prefix of a path in $\Sigma \cup \{\sigma\}$. Hence $\mathbb{T}_1$ will still satisfy Σ and violate σ. Next, consider $\mathbb{P}' - \mathbb{P}$. If we let $P = \{p_1, \ldots, p_m\}$ be the paths in $\mathbb{P}' - \mathbb{P}$, plus all the prefix paths of the paths in $\mathbb{P}' - \mathbb{P}$ except ρ, then we order the paths

in P so that if $i < j$ then $p_j \not\succ p_i$. We then add in sequence a new node to $\mathbb{T}_1$ for every path $p \in P$, and if there is a path $p' \in P$ such that $p' = \mathit{Parent}(p)$, then we make the node for p a child of the node for p'. If we call the resulting tree $\mathbb{T}'$, then it is clear that $\mathbb{T}' \models \Sigma$ but $\mathbb{T}' \not\models \sigma$ since $\mathbb{T}_1$ has the same properties, and $\mathbb{T}' \models \mathbb{P}'$ and is complete since $\mathbb{T}$ is complete and from the construction procedure. Thus $(\mathbb{P}', \Sigma) \not\vdash \sigma$, and the converse holds from the same argument. □

3.1. Soundness of the axiom system

In this section we establish the first main result of this article, namely that the axiom system defined in the previous section is sound, that is if σ can be derived from $(\mathbb{P}, \Sigma)$ using our axiom system, then $(\mathbb{P}, \Sigma) \vdash \sigma$. To show this, we first formalize the notion of a derivation [1].

Definition 18. Denote by $\mathcal{A}$ Axioms A1–A8, and suppose that $\Sigma \models \mathbb{P}$ and $\sigma \models \mathbb{P}$. Σ *derives* σ (using $\mathcal{A}$), denoted by $(\mathbb{P}, \Sigma) \stackrel{d}{\vdash} \sigma$, if there exists a sequence of XFDs $\sigma_1, \ldots, \sigma_n = \sigma$, $n \geqslant 1$, such that for all $i \in [1, n]$ either:

(i) $\sigma_i \in \Sigma$; or
(ii) there is a substitution for some rule $\tau \in \mathcal{A}$ such that σ_i corresponds to a consequent of τ, and such that every XFD in the antecedent of τ is in the set $\{\sigma_j | j \in [1, i-1]\}$.

We next establish two preliminary lemmas. The first lemma presents some basic properties of $\cap$ and $\succeq$ (Definition 4). The proof is omitted since it follows directly from the definitions.

Lemma 19. *Let $\mathbb{P}$ be a set of paths and let p, q, and r be arbitrary paths in $\mathbb{P}$.*

(i) *If $p \succ (q \cap r)$ then $p \succ q$, and if $p \succeq (q \cap r)$ then $p \succeq q$.*
(ii) *If $p \succeq q$ and $p \succeq r$ then $p \succeq (q \cap r)$.*
(iii) *For every pair of paths p and q, $p = (p \cap q) * z_1$ and $q = (p \cap q) * z_2$, where z_1 and z_2 are label-sequences (possibly empty) such that $z_1 \cap z_2 = \epsilon$.*
(iv) *$p = p \cap q$ iff $p \succeq q$.*
(v) *If $p \succeq q$ and $q \succeq p$ then $p = q$.*
(vi) *If $p \succeq r$ and $q \succeq r$, then either $p \succeq q$ or $q \succeq p$.*
(vii) *$p \cap q \succeq p$.*
(viii) *$p \cap p = p$.*
(ix) *If $p \succeq q$ then $p \cap r \succeq q \cap r$.*
(x) *$p \cap (q \cap r) = (p \cap q) \cap r$.*
(xi) *If $p \succeq q$ and $q \succeq r$ then $p \succeq r$.*

Next, we give some basic properties of the *closest* function and the *aancestor* function (Definition 9). Once again, the proof is omitted since it follows directly from the definition. In the lemma, recall that in the definition of *closest*, x_j^i is the node in $[\![p_i \cap p_j]\!]$ that is an *aancestor* of both v_i and v_j.

Lemma 20. *If $\mathbb{P}$ is a set of paths and $\{p, p_i, p_j\} \subseteq \mathbb{P}$, and $v_i, \bar{v}_i, v_j, \bar{v}_j, v_k$ are nodes in an XML tree $\mathbb{T}$ such that $\mathbb{T} \models \mathbb{P}$, then the following hold*:

(i) *$\mathit{closest}(v_i, v_j) = \mathit{closest}(v_j, v_i)$.*
(ii) *$\mathit{closest}(v_i, v_i)$ is true.*
(iii) *If $\mathit{closest}(v_i, v_j)$ is true and $v_i, v_j \in [\![p]\!]$ for some path $p \in \mathbb{P}$, then $v_i = v_j$.*
(iv) *If $v_i \in [\![p_i]\!]$ and $v_j \in [\![p_j]\!]$ and $p_i \succeq p_j$ and $\mathit{closest}(v_i, v_j)$ is true, then $x_j^i = v_i$ and $v_i \in \mathit{aancestor}(v_j)$.*
(v) *If $v_i \in [\![p_i]\!]$ and $v_j \in [\![p_i]\!]$ and $v_i \in \mathit{aancestor}(v_k)$ and $v_j \in \mathit{aancestor}(v_k)$, then $v_i = v_j$.*
(vi) *If $\{v_i, \bar{v}_i\} \subseteq [\![p_i]\!]$ and $\{v_j, \bar{v}_j\} \subseteq [\![p_j]\!]$ and $\mathit{closest}(v_i, v_j)$ is true and $\mathit{closest}(\bar{v}_i, \bar{v}_j)$ is true and $v_i \neq \bar{v}_i$ and $v_j = \bar{v}_j$, then $p_i \not\succeq p_j$.*
(vii) *If $v_j \in [\![p_j]\!]$ and $p_i \succeq p_j$, then there exists exactly one node v_i such that $v_i \in [\![p_i]\!]$ and $v_i \in \mathit{aancestor}(v_j)$.*
(viii) *If $v_i \in \mathit{aancestor}(v_k)$ and $v_i \in [\![p_i]\!]$, and $v_j \in \mathit{aancestor}(v_k)$ and $v_j \in [\![p_j]\!]$ and $p_i \succeq p_j$ then, $v_i \in \mathit{aancestor}(v_j)$.*
(ix) *If $v_i \in \mathit{aancestor}(v_j)$ and $v_j \in \mathit{aancestor}(v_k)$, then $v_i \in \mathit{aancestor}(v_k)$.*
(x) *If $v_i \in \mathit{aancestor}(v_j)$ and $v_i \in [\![p_i]\!]$ and $v_j \in [\![p_j]\!]$, then $p_i \succeq p_j$.*
(xi) *If $v_i \in \mathit{aancestor}(v_j)$, then $\mathit{closest}(v_i, v_j)$ is true.*

The following lemma shows that given a set of paths $p_1, \ldots, p_n$ and a set of nodes defined over the paths for which *closest* is true for every pair of paths, and another path p_{n+1}, it is always possible to find a node in $[\![p_{n+1}]\!]$ such that the nodes defined over the paths in $p_1, \ldots, p_n, p_{n+1}$ also satisfy the *closest* property pairwise.

Lemma 21. *For every subset $p_1, \ldots, p_n, p_{n+1}$ of a set of paths $\mathbb{P}$, where $n \geqslant 1$, if there exist nodes $v_1 \in [\![p_1]\!], \ldots, v_n \in [\![p_n]\!]$ in a complete XML tree $\mathbb{T}$ such that $v_1, \ldots, v_n$ is a cn-set, then there exists a node $v_{n+1} \in [\![p_{n+1}]\!]$ such that $v_1, \ldots, v_n, v_{n+1}$ is a cn-set.*

Proof. See Appendix A. □

Next, in several of the proofs in the rest of the article we frequently need to list the conditions under which an XFD is not logically implied. The following definition is a direct consequence of the definition of XFD satisfaction (Definition 11) and logical implication (Definition 14).

Definition 22. Let Σ be a set of XFDs, let σ be the XFD $p_1, \ldots, p_n \rightarrow p_{n+1}, \ldots, p_{n+m}$ and let $\mathbb{P}$ be a set of paths such that $\Sigma \cup \{\sigma\} \models \mathbb{P}$. Then $(\mathbb{P}, \Sigma) \nvdash \sigma$ if there exists an XML tree $\mathbb{T}$ such that:

(i) $\mathbb{T} \models \mathbb{P}$;
(ii) $\mathbb{T} \models \Sigma$;
(iii) $\mathbb{T} \not\models \sigma$, that is there exist nodes $v_1, v'_1 \in [\![p_1]\!], \ldots, v_{n+m},\ v'_{n+m} \in [\![p_{n+m}]\!]$ in $\mathbb{T}$, where $n \geqslant 1$ and $m \geqslant 1$, such that the following hold:
  (a) $v_1, \ldots, v_{n+m}$ and $v'_1, \ldots, v'_{n+m}$ are cn-sets;
  (b) $\forall i \in [1, n]$, $val(v_i) = val(v'_i)$;
  (c) $\exists j \in [n+1, n+m]$ such that $val(v_j) \neq val(v'_j)$.

Theorem 23. *Axioms* A1–A8 *are sound for the implication of XFDs in complete XML trees.*

Proof. *Axiom* A1. The proof is by contra-positive, and so assume that $(\mathbb{P}, \Sigma) \nvdash \sigma$, where $\sigma = p_1, \ldots, p_n \rightarrow p_i$. So from Definition 22 there are nodes v, v' in $[\![p_i]\!]$ in $\mathbb{T}$ such that $val(v) \neq val(v')$ (since p_i is on the r.h.s.), and nodes v_i, v'_i in $[\![p_i]\!]$ such that $closest(v_i, v)$ is true, $closest(v'_i, v')$ is true and $val(v_i) = val(v'_i)$ (since p_i is on the l.h.s.). However, from Lemma 20 (iii), $v_i = v$ and $v'_i = v'$. This is a contradiction since $val(v) \neq val(v')$ and $val(v_i) = val(v'_i)$ and so we conclude that $\Sigma \vdash \sigma$.

Axiom A2. The proof is by contra-positive, and so we show that if $(\mathbb{P}, \Sigma) \nvdash p_0, p_1, \ldots, p_n \rightarrow p_0, q_1, \ldots, q_m$, then $(\mathbb{P}, \Sigma) \nvdash p_1, \ldots, p_n \rightarrow q_1, \ldots, q_m$. Let us define $p_{n+1} = p_0, p_{n+2} = q_1, \ldots, p_{n+m+1} = q_m$. Considering (c) of Definition 22, suppose first that $j = n+1$. Then using the same arguments as in A1, if $closest(v_0, v_{n+1})$ is true then $v_{n+1} = v_0$, and if $closest(v'_0, v'_{n+1})$ is true then $v'_{n+1} = v'_0$. However this is a contradiction since $val(v_0) = val(v'_0)$ by (b) and so since $val(v_{n+1}) = val(v'_{n+1})$, but $val(v_{n+1}) \neq val(v'_{n+1})$ from (c). So $j \neq n+1$, and thus $j = n+k$ for some $k > 1$. Then by considering the nodes $v_1, \ldots, v_n, v_{n+k}$ and $v'_1, \ldots, v'_n, v'_{n+k}$, we have that for all $i \in [1, n]$, $val(v_i) = val(v'_i)$ from (i) and $val(v_{n+k}) \neq val(v'_{n+k})$ from (iii) and so $\mathbb{T} \not\models p_1, \ldots, p_n \rightarrow q_1, \ldots, q_m$ and so $(\mathbb{P}, \Sigma) \nvdash p_1, \ldots, p_n \rightarrow q_1, \ldots, q_m$ as claimed.

Axiom A3. The proof is by contra-positive, and so we show that if $(\mathbb{P}, \Sigma) \nvdash p_1, \ldots, p_n \rightarrow s$, then either $(\mathbb{P}, \Sigma) \nvdash p_1, \ldots, p_n \rightarrow q_1, \ldots, q_m$ or $(\mathbb{P}, \Sigma) \nvdash q_1, \ldots, q_m \rightarrow s$. Let $p_{n+1} = s$. Since $(\mathbb{P}, \Sigma) \nvdash p_1, \ldots, p_n \rightarrow s$, there exists an XML tree $\mathbb{T}$ satisfying Definition 22. So since $closest(v_i, v_j)$ is true for all $i, j \in [1, n+1]$ then, by repeatedly applying Lemma 21 and considering the set of paths $p_1, \ldots, p_{n+1}, q_1 \ldots, q_m$, there exist nodes $v_{n+2}, \ldots, v_{n+m+1}$ in $\mathbb{T}$ such that $v_{n+2} \in [\![q_1]\!], \ldots, v_{n+m+1} \in [\![q_m]\!]$ and $closest(v_i, v_j)$ is true for all $i, j \in [1, n+m+1]$. For the same reasons, there exist nodes $v'_{n+2}, \ldots, v'_{n+m+1}$ in $\mathbb{T}$ such that $v'_{n+2} \in [\![q_1]\!], \ldots, v'_{n+m+1} \in [\![q_m]\!]$ and $closest(v'_i, v'_j)$ is true for all $i, j \in [1, n+m+1]$.

There are then two possibilities. The first is that for all $i \in [n+2, n+m+1]$, $val(v_i) = val(v'_i)$. In this case, $\mathbb{T} \not\models q_1, \ldots, q_m \rightarrow s$ because $val(v_{n+1}) \neq val(v'_{n+1})$. The other possibility is that there exists $j \in [n+2, n+m+1]$ such that $val(v_j) \neq val(v'_j)$. In this case $\mathbb{T} \not\models p_1, \ldots, p_n \rightarrow q_1, \ldots, q_m$, which establishes A3.

Axiom A4. The argument is again by contra-positive, and so let $\mathbb{T}$ be a tree in Definition 22 such that $\mathbb{T} \not\models p_1, \ldots, p_n \rightarrow q$. Then there exist nodes v_1, v'_1 in $[\![p]\!]$ and nodes v_2, v'_2 in $[\![q]\!]$ such that $closest(v_1, v_2)$ and $closest(v'_1, v'_2)$ are true and $val(v_1) = val(v'_1)$ and $val(v_2) \neq val(v'_2)$. Since for all $i \in [1, n]$, $p_i \cap q = \rho$ from the axiom, then for every node $v_i \in [\![p_i]\!]$, where $i \in [1, n]$, $closest(v_i, v_2)$ and $closest(v_i, v'_2)$ are true because v_i, v_2 and v'_2 must all be descendants of v_ρ and $v_\rho \in [\![p_i \cap q]\!]$. So $\mathbb{T} \not\models p_1, \ldots, p_n \rightarrow q$, which establishes A4.

Axiom A5. The proof of this axiom is much more complex than the other axioms because of the large number of different cases that need to be considered. Accordingly, we present a detailed proof in Appendix A but also give here an outline of the proof to assist the reader.

The argument is again by contra-positive, and so let $\mathbb{T}$ be a tree in Definition 22 such that $\mathbb{T} \not\models p \rightarrow q$. We first note that from Axiom A1, we can assume that for all $i \in [1, n]$, $q \neq p_i$ and we let $p_{n+1} = q$.

The first step in the proof is to modify the set of nodes $\{v'_1, \ldots, v'_n\}$ as follows. For any node v_i in the set $\{v_1, \ldots, v_n\}$, if $closest(v_i, v'_{n+1})$ is also true, then replace v'_i by v_i in the set $\{v'_1, \ldots, v'_n\}$. We can then show that this modified set of nodes is also a cn-set.

The next step in the proof is to show that it is always possible to choose two other sets of nodes $\{\bar{v}_1, \ldots, \bar{v}_{n+1}\}$ and $\{\bar{v}'_1, \ldots, \bar{v}'_{n+1}\}$ from $\mathbb{T}$ so that $\mathbb{T} \not\models p'_1, \ldots, p'_n \rightarrow p_{n+1}$. To be more precise, $\{\bar{v}_1, \ldots, \bar{v}_{n+1}\}$ and $\{\bar{v}'_1, \ldots, \bar{v}'_{n+1}\}$ must satisfy the following conditions:

(a′) $\bar{v}_1 \in [\![p'_1]\!], \ldots, \bar{v}_{n+1} \in [\![p_{n+1}]\!]$ and $\bar{v}'_1 \in [\![p'_1]\!], \ldots, \bar{v}'_{n+1} \in [\![p_{n+1}]\!]$;
(b′) $\{\bar{v}_1, \ldots, \bar{v}_{n+1}\}$ is a cn-set;
(c′) $\{\bar{v}'_1, \ldots, \bar{v}'_{n+1}\}$ is a cn-set;
(d′) $\forall i \in [1, n], val(\bar{v}_i) = val(\bar{v}'_i)$;
(e′) $val(\bar{v}_{n+1}) \neq val(\bar{v}'_{n+1})$.

This construction procedure takes a number of steps, which we now outline.

Step 1: We let $\bar{v}_{n+1} = v_{n+1}$ and $\bar{v}'_{n+1} = v'_{n+1}$.

Step 2: We choose for every path $p'_i \in \{p'_1, \ldots, p'_n\}$, an associated maximal path, denoted by $M(p'_i)$, that is defined to be the path p that maximizes $p'_i \cap p$ w.r.t. $\succeq$, that is

$$\forall p \in \{p_1, \ldots, p_{n+1}, p'_1, \ldots, p'_{i-1}\}, p'_i \cap p \succeq p'_i \cap M(p'_i). \tag{1}$$

If there is more than one such maximal path, then the choice of $M(p'_i)$ is arbitrary.

Step 3: We show that for all $i \in [1, n]$, one of the following conditions must hold:

(i) $p_i = p'_i$;
(ii) $p_i \succ p'_i$;
(iii) $p_i \succ p_{n+1}$.

Step 4: We choose nodes in increasing index order as follows. For each $i \in [1, n]$, choose nodes $\bar{v}_i$ and $\bar{v}'_i$ as follows:

(A) If (i) from Step 3 holds, then $\bar{v}_i = v_i$ and $\bar{v}'_i = v'_i$.
(B) If (ii) from Step 3 holds, let $\bar{v}_i$ be any node in $[\![p'_i]\!]$ such that $closest(\bar{v}_i, v_m)$ is true, where v_m is the node in the cn-set $\{v_1, \ldots, v_{n+1}\}$ that is also in $[\![M(p'_i)]\!]$, and also let $\bar{v}'_i = \bar{v}_i$.
(C) If (iii) from Step 3 holds, let $\bar{v}_i$ be any node in $[\![p'_i]\!]$ such that $closest(\bar{v}_i, v_m)$ is true, where v_m is the node in the cn-set $\{v_1, \ldots, v_{n+1}\}$ that is also in $[\![M(p'_i)]\!]$ (as in (ii)), and let $\bar{v}'_i$ be any node in $[\![p'_i]\!]$ such that $closest(\bar{v}'_i, v'_m)$ is true, where v'_m is the node in the cn-set $\{v'_1, \ldots, v'_{n+1}\}$ that is also in $[\![M(p'_i)]\!]$.

We now demonstrate Step 4 by an example.

Example 24. Consider the XFD $\sigma_1 = \rho.\mathtt{A.B.C}, \rho.\mathtt{A.B.C.@C}, \rho.\mathtt{A.D.E.@E} \rightarrow \rho.\mathtt{A.D.F.@F}$ which from Axiom A5 implies the XFD $\sigma_2 = \rho.\mathtt{A.B}, \rho.\mathtt{A.B.C.@C}, \rho.\mathrm{A.D.F} \rightarrow \rho.\mathtt{A.D.F.@F}$. In Fig. 6, σ_2 is violated when we consider the cn-sets $\{v_1, v_2, v_3, v_4\}$ and $\{v_1, v_2, v'_3, v'_4\}$.

Let $\rho.\mathtt{A.B.C} = p'_1, \rho.\mathtt{A.B.C.@C} = p'_2, \rho.\mathtt{A.D.E.@E} = p'_3, \rho.\mathtt{A.D.F.@F} = p_4 = q$ and $\rho.\mathtt{A.B} = p_1, \rho.\mathtt{A.B.C.@C} = p_2, \rho.\mathtt{A.D.F} = p_3$.

We then proceed as follows. For $i = 1$, (B) applies. Hence $M(p'_1) = p_2$ and so $v_m = v_2$ and the node in $v \in [\![p'_1]\!]$ that satisfies $closest(\bar{v}_1, v_m)$ is $v = \bar{v}_1$, and we also let $\bar{v}'_1 = \bar{v}_1$. For $i = 2$, (A) applies and so we let $\bar{v}_2 = v_2$ and $\bar{v}'_2 = v_2$. For $i = 3$, (C) applies and so we compute $M(p'_3) = p_3$ (or p_4) and so $v_m = v_3$ and $v'_m = v'_3$. Hence $\bar{v}'_3 = \bar{v}_3$ and one can verify that the cn-sets $\{\bar{v}_1, v_2, \bar{v}_3, v_4\}$ and $\{\bar{v}_1, v_2, \bar{v}_3, v'_4\}$ cause σ_2 to be violated.

Axiom A6. The argument is again by contra-positive, and so let $\mathbb{T}$ be a tree in Definition 22 such that $\mathbb{T} \not\models p \rightarrow q$. Thus there exist nodes $v, v' \in [\![p]\!]$ and nodes $\bar{v}, \bar{v}' \in [\![q]\!]$ in $\mathbb{T}$ such that $closest(v, \bar{v})$ is true, $closest(v', \bar{v}')$ is true, $val(v) = val(v')$ and $val(\bar{v}) \neq val(\bar{v}')$. However, since $q \succ p$ and $closest(v, \bar{v})$ is true, from Lemma 20 (iv) we derive that $\bar{v} \in aancestor(v)$. Similarly, $\bar{v}' \in aancestor(v')$. However, since $Last(p) \in \mathbf{E}$ and using the definition of val, if $val(v) = val(v')$ then $v = v'$. Also, since $\mathbb{P}$ is downward closed and $q \succ p$, $Last(q) \in \mathbf{E}$ and so $\bar{v} \neq \bar{v}'$ because $val(\bar{v}) \neq val(\bar{v}')$. Hence v has two ancestor nodes, $\bar{v}$ and $\bar{v}'$, which contradicts the fact that $\mathbb{T}$ is a tree, and so $\mathbb{T} \models p \rightarrow q$.

Axiom A7. The argument is again by contra-positive, and so let $\mathbb{T}$ be a tree in Definition 22 such that $\mathbb{T} \not\models Parent(p) \rightarrow p$. So there exist nodes $v, v' \in [\![Parent(p)]\!]$ and $\bar{v}, \bar{v}' \in [\![p]\!]$ such that $closest(v, \bar{v})$ is true, $closest(v', \bar{v}')$ is true, $val(v) = val(v')$ and $val(\bar{v}) \neq val(\bar{v}')$. By definition of $Parent$, $Parent(p) \succ p$, and so if $closest(v, \bar{v})$ is true then $v = parent(\bar{v})$. Similarly, $v' = parent(\bar{v}')$. Since $Last(Parent(p)) \in \mathbf{E}$ by definition, then $v = v'$ because $val(v) = val(v')$. However, since $val(\bar{v}) \neq val(\bar{v}')$, this implies that $\bar{v}$ and $\bar{v}'$ are distinct attribute children of v with the same label, which contradicts the definition of an XML tree and so $\mathbb{T} \models Parent(p) \rightarrow p$.

Axiom A8. This is automatic since there is only one node, v_ρ, in $[\![\rho]\!]$ and so every XFD with ρ on the r.h.s. is automatically satisfied. □

4. Completeness of the axiom system

In this section we establish the second main result of this article, namely that our axiom system defined in the previous section is complete. Since the proof of this fact requires several stages, we first present an outline of the steps involved.

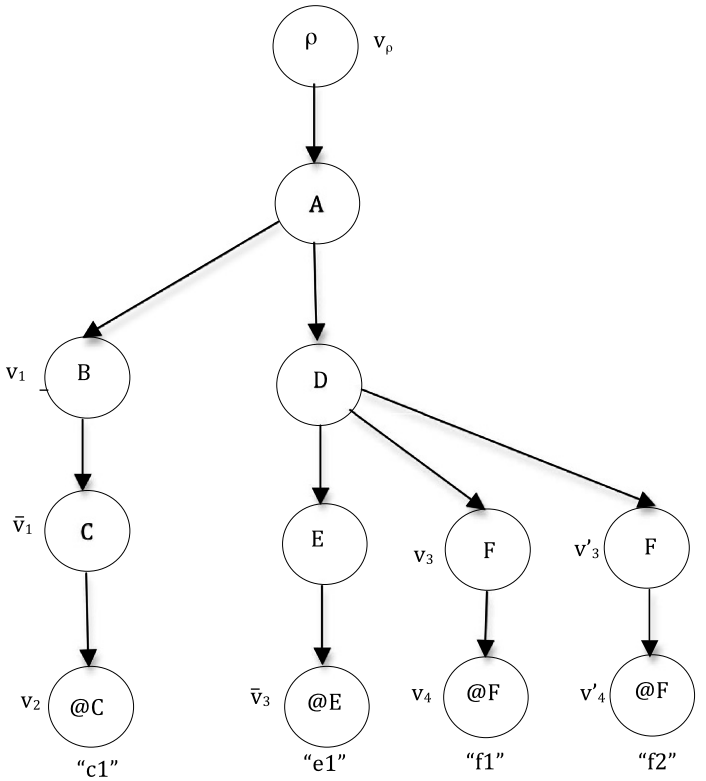

Fig. 6. An XML tree illustrating the proof of Axiom A5.

- We first introduce the notion of a *text XFD*, which is one where every path on the l.h.s. of the XFD ends in an attribute or text label. Our motivation for considering this restricted type of XFD is that the implication problem for such XFDs is easier to solve than for arbitrary XFDs, which can have paths ending in element labels on the l.h.s. of an XFD, and so we solve the implication problem for a set of arbitrary XFDs by showing that it can be reduced to solving the implication problem for an equivalent set of text XFDs.
  This reduction process takes several sub-steps. First, we present a procedure that takes an arbitrary set of XFDs Σ as input and outputs a set of text XFDs, $\alpha(\Sigma)$, by introducing a new identifying attribute label for every path in the l.h.s. of an XFD in Σ that ends with an element label. In the next sub-step we prove that Σ implies another XFD σ if and only if $\alpha(\Sigma)$ implies $\alpha(\sigma)$ (Theorem 31), where $\alpha(\sigma)$ is the text XFD corresponding to σ. Finally, we show that if $\alpha(\sigma)$ can be derived from $\alpha(\Sigma)$ using our XFD axiom system, then σ can also be derived from Σ using our axiom system (Theorem 33).
- The next step in the proof is to show that our axiom system is complete for sets of text XFDs. To do this, we define a closure algorithm for XFDs (Algorithm 1) and show that the algorithm is sound and complete for sets of text XFDs (Theorem 38). We do this by introducing a chase procedure (Algorithm 2 and Algorithm 3) for XFDs that is an extension of the classical chase algorithm for relational dependencies, but applies to XML trees rather than relations. We then use the soundness and completeness of the closure algorithm for text XFDs to show that our XFD axiom system is complete for sets of text XFDs (Corollary 39).
- The final step in the proof involves combining the previous two steps to convert the implication and derivation problems for sets of arbitrary XFDs to the same problems for sets of text XFDs. This leads to the main result of the article (Theorem 41), which shows that our XFD axiom system is sound and complete for sets of arbitrary XFDs. As a corollary, we show that the implication problem for an arbitrary set of XFDs can be computed by first converting the set to an equivalent set of text XFDs, and then using either the closure algorithm, or the chase algorithm.

4.1. Text XFDs

We first define some text related concepts.

Definition 25. We are given a set of paths $\mathbb{P}$, a set of XFDs Σ and an XML tree $\mathbb{T}$ such that $\Sigma \models \mathbb{P}$, $\mathbb{T} \models \mathbb{P}$ and $\mathbb{T} \models \Sigma$.

- A path $p \in \mathbb{P}$ is a *text path* if $Last(p) \notin \mathbf{E}$, and $\mathbb{P}$ is a set of text paths if every path in $\mathbb{P}$ is a text path.

- An XFD $\sigma \in \Sigma$ is a *text XFD* if every path on the l.h.s. of σ is a text path, and Σ is a set of text XFDs if every XFD in Σ is a text XFD.
- $\mathbb{T}$ is a *text XML tree* if for every leaf node $v \in \mathbb{T}$, $lab(v) \notin \mathbf{E}$.

We note that our definition of a text XFD only requires that the paths on the l.h.s. of the XFD are text paths, it does not require that the path on the r.h.s. be a text path. The reason for this difference is that it is only text paths on the l.h.s. of an XFD that cause technical problems in solving the implication problem, hence our definition of a text XFD only requires that such paths not be present.

We now consider the problem of how to convert an arbitrary set of XFDs to an equivalent set of text XFDs. Our solution to this problem extends the classical concept of a surrogate in a relational database [1], where an artificial attribute is introduced as a key into a relational schema. Given an arbitrary XFD, we replace every path on the l.h.s. of the XFD that ends with an element label by extending the path with a new attribute label so that the new path is an identifier for the old path.

To demonstrate this idea, consider a unary XFD $p \to q$, where $Last(p) \in \mathbf{E}$. We introduce a new attribute label, denoted by $l_@$, such that the new path $p * l_@$ uniquely identifies p nodes in any XML tree, that is the XFD $p * l_@ \to p$ holds. If we then replace the set $\{p \to q\}$ by the set $\{p * l_@ \to p, p * l_@ \to q\}$, the new set and the old set are equivalent. This is because the XFD $p \to q$ is implied by $\{p * l_@ \to p, p * l_@ \to q\}$ since by Axiom A7 $p \to p * l_@$ and then applying Axiom A3 shows that $p \to q$. For the reverse direction, $p * l_@ \to p$ holds by assumption and if we combine this with $p \to q$ and apply Axiom A3 then we derive $p * l_@ \to q$.

We now formalize the procedure just illustrated, using the symbol α to denote this conversion to text.

Definition 26. Let $\mathbb{P}$ be a set of paths, $p \in \mathbb{P}$, Σ a set of XFDs, $\sigma \in \Sigma$ and $\mathbb{T}$ an XML tree. Suppose also that $\Sigma \models \mathbb{P}$, $\mathbb{T} \models \mathbb{P}$ and $\mathbb{T} \models \Sigma$. First, add a new label $l_@$ to the set of attributes $\mathbf{A}$ which, without any loss of generality, we assume not to already be in $\mathbf{A}$.

- $ele(\mathbb{P}) \stackrel{\text{def}}{=} \{p \mid p \in \mathbb{P}$ and $Last(p) \in \mathbf{E}$ and p appears on the l.h.s. of an XFD in $\Sigma\}$.
- $\alpha(p) \stackrel{\text{def}}{=} p * l_@$ if $p \in ele(\mathbb{P})$, else $\alpha(p) \stackrel{\text{def}}{=} p$.
- $\alpha(\mathbb{P}) \stackrel{\text{def}}{=} \mathbb{P} \cup \{\alpha(p) \mid p \in ele(\mathbb{P})\}$.
- If σ is the XFD $p_1, \ldots, p_n \to p_{n+1}$, then we define $\alpha(\sigma)$ to be the XFD $\alpha(p_1), \ldots, \alpha(p_n) \to \alpha(p_{n+1})$.
- $\alpha(\Sigma) \stackrel{\text{def}}{=} \{\alpha(\sigma) \mid \sigma \in \Sigma\} \cup \{\alpha(p) \to p \mid p \in ele(\mathbb{P})\}$.
- We augment $\mathbb{T}$ to create $\alpha(\mathbb{T})$ as follows. For each $p \in ele(\mathbb{P})$ and each node $v \in [\![p]\!]$ in $\mathbb{T}$, add a new attribute child node of v with label $l_@$ and assign a *unique* (but arbitrary) *val* to it.

We note that it follows immediately from this definition that $\alpha(\Sigma) \models \alpha(\mathbb{P})$, $\mathbb{P} \subseteq \alpha(\mathbb{P})$, $\alpha(\mathbb{T})$ is complete and $\alpha(\mathbb{T}) \models \alpha(\mathbb{P})$. We now illustrate this definition by an example.

Consider the example in Fig. 1 and let:

$$l_\alpha = \texttt{@\#};$$

$$\mathbb{P} = \{\rho,\ \rho\texttt{.Dept},\ \rho\texttt{.Dept.Section},\ \rho\texttt{.Dept.Section.Emp},\ \rho\texttt{.Class},$$
$$\rho\texttt{.Dept.Section.Emp.S},\ \rho\texttt{.Dept.Section.@Name},\ \rho\texttt{.Class.@Id}\}$$

$$\Sigma = \{\rho\texttt{.Dept.Section.@Name} \to \rho\texttt{.Class},\ \rho\texttt{.Class},\ \rho\texttt{.Dept.Section.Emp.S} \to \rho\texttt{.Dept}\}.$$

Then:

$$ele(\mathbb{P}) = \{\rho\texttt{.Class}\};$$

$$\alpha(\mathbb{P}) = \{\rho,\ \rho\texttt{.Dept},\ \rho\texttt{.Dept.Section},\ \rho\texttt{.Dept.Section.Emp},\ \rho\texttt{.Class},\ \rho\texttt{.Class.@\#},$$
$$\rho\texttt{.Dept.Section.Emp.S},\ \rho\texttt{.Dept.Section.@Name},\ \rho\texttt{.Class.@Id}\};$$

$$\alpha(\Sigma) = \{\rho\texttt{.Dept.Section.@Name} \to \rho\texttt{.Class.@\#},\ \rho\texttt{.Class.@\#}\ ,$$
$$\rho\texttt{.Dept.Section.Emp.S} \to \rho\texttt{.Dept},\ \rho\texttt{.Class.@\#} \to \rho\texttt{.Class}\};$$

and if we denote the XML tree in Fig. 1 by $\mathbb{T}$, then $\alpha(\mathbb{T})$ is shown in Fig. 7. In this example the only path in $\mathbb{P}$ that is changed is the path $\rho\texttt{.Class}$.

We next show that using this conversion process, we can convert the implication problem for an arbitrary set of XFDs to the implication problem for a set of text XFDs. To do this we first need to define the notion of subsumption between two XML trees, and we then establish a lemma which shows that if an XML tree $\mathbb{T}_1$ is subsumed by an XML tree $\mathbb{T}_2$ and $\mathbb{T}_1$ violates an XFD, then $\mathbb{T}_2$ must also violate the XFD.

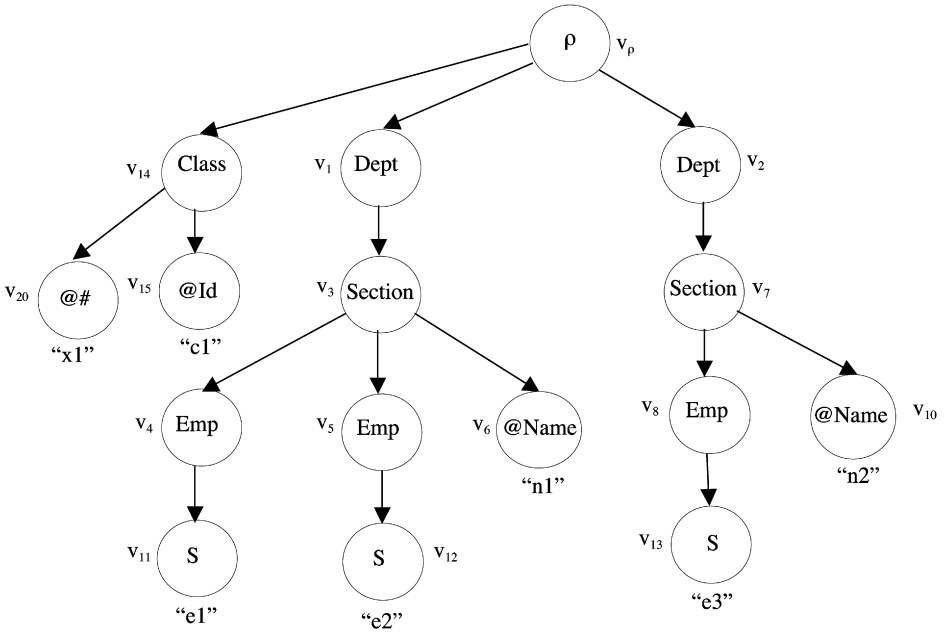

Fig. 7. A text XML tree.

Definition 27. Given two XML trees $\mathbb{T}_1 = (V_1, lab_1, ele_1, att_1, val_1, \rho, v_{\rho_1})$ and $\mathbb{T}_2 = (V_2, lab_2, ele_2, att_2, val_2, \rho, v_{\rho_2})$, $\mathbb{T}_1$ is subsumed by $\mathbb{T}_2$, denoted by $\mathbb{T}_1 \sqsubseteq \mathbb{T}_2$, iff:

(i) $V_1 \subseteq V_2$;
(ii) the functions lab_1 and lab_2 and val_1 and val_2 are identical on V_1;
(iii) for all nodes $v, v' \in V_1$, v is the parent of v' in $\mathbb{T}_1$ iff v is the parent of v' in $\mathbb{T}_2$.

Lemma 28. *Let $\mathbb{P}_1$ and $\mathbb{P}_2$ be sets of paths such that $\mathbb{P}_1 \subseteq \mathbb{P}_2$, let $\mathbb{T}_1 = (V_1, lab_1, ele_1, att_1, val_1, v_{\rho_1})$ be an XML tree such that $\mathbb{T}_1 \models \mathbb{P}_1$ and let $\mathbb{T}_2 = (V_2, lab_2, ele_2, att_2, val_2, v_{\rho_2})$ be an XML tree such that $\mathbb{T}_2 \models \mathbb{P}_2$ and $\mathbb{T}_1 \sqsubseteq \mathbb{T}_2$, and let σ be an XFD. If $\mathbb{T}_1 \not\models \sigma$ then $\mathbb{T}_2 \not\models \sigma$.*

Proof. Let σ be the XFD $p_1, \ldots, p_n \to p_{n+1}, \ldots, p_{n+m}$, and let $v_1, v'_1, \ldots, v_{n+m}, v'_{n+m}$ be as defined in Definition 22. We need to show that (iii) of Definition 22 also holds in $\mathbb{T}_2$.

Consider (a) of Definition 22. From the definition of subsumption, the nodes $v_1, v'_1, \ldots, v_{n+m}, v'_{n+m}$ are also nodes in $\mathbb{T}_2$. Let v_i and v_j be arbitrary nodes in $v_1, \ldots, v_{n+m}$ and let v'_i and v'_j be arbitrary nodes in $v'_1, \ldots, v'_{n+m}$. Since $closest(v_i, v_j)$ is true in $\mathbb{T}_1$ from (a) of Definition 22, there exists a node $x^i_j \in [\![p_i \cap p_j]\!]$ in $\mathbb{T}_1$ such that $x^i_j \in aancestor(v_i)$ and $x^i_j \in aancestor(v_j)$. Similarly, since $closest(v'_i, v'_j)$ is true in $\mathbb{T}_1$, there exists a node $x'^i_j \in [\![p_i \cap p_j]\!]$ such that $x'^i_j \in aancestor(v'_i)$ and $x'^i_j \in aancestor(v'_j)$. So by property (iii) of subsumption, $x^i_j \in aancestor(v_i)$, and $x^i_j \in aancestor(v_j)$ in $\mathbb{T}_2$ and $x'^i_j \in aancestor(v'_i)$ and $x'^i_j \in aancestor(v'_j)$ in $\mathbb{T}_2$. So we deduce that (a) of Definition 22 holds in $\mathbb{T}_2$, and (b) and (c) hold by property (ii) of subsumption. Thus $\mathbb{T}_2 \not\models \sigma$ as required. □

Next, we establish an important preliminary theorem which shows that an XML tree $\mathbb{T}$ satisfies an XFD if and only if $\alpha(\mathbb{T})$ satisfies the corresponding text XFD. To do this we need a preliminary lemma, illustrated in Fig. 8.

Lemma 29. *If $\mathbb{P}$ is a set of paths and $\mathbb{T}$ an XML tree such that $\mathbb{T} \models \mathbb{P}$ and there exist paths p_1, p'_1, p_2, p'_2 in $\mathbb{P}$ and nodes v_1, v'_1, v_2, v'_2 in $\mathbb{T}$ where*:

(i) $v_1 \in [\![p_1]\!]$, $v'_1 \in [\![p'_1]\!]$, $v_2 \in [\![p_2]\!]$, $v'_2 \in [\![p'_2]\!]$; *and*
(ii) $p_1 = Parent(p'_1)$, $v_1 = parent(v'_1)$, $p_2 = Parent(p'_2)$, $v_2 = parent(v'_2)$; *and*
(iii) $\{lab(v'_1), lab(v'_2)\} \subseteq \mathbf{A} \cup \{S\}$;
(iv) $p'_1 \neq p'_2$;

then $closest(v_1, v_2) = closest(v'_1, v'_2) = closest(v_1, v'_2)$.

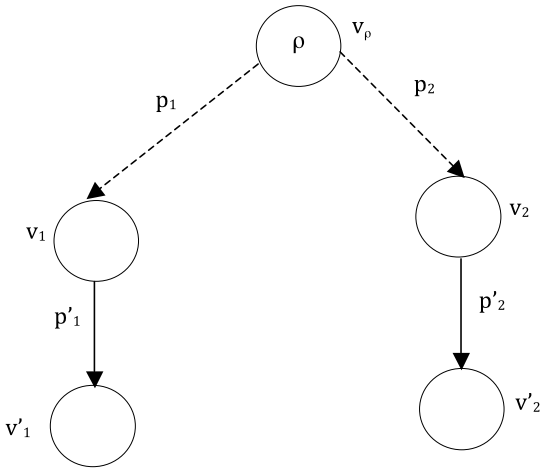

Fig. 8. An XML tree illustrating Lemma 29.

Proof. We shall show that $closest(v'_1, v'_2) \iff closest(v_1, v'_2)$ and $closest(v'_1, v'_2) \iff closest(v_1, v_2)$.

We first observe that because of (iii) and (iv), the only case that needs to be considered is when $p'_1 \not\succeq p'_2$ and $p'_2 \not\succeq p'_1$. We next claim that the following holds:

$$p_1 \cap p_2 = p'_1 \cap p'_2. \tag{2}$$

To show this, since $p_1 \succeq p'_1$ and $p_2 \succeq p'_2$ by condition (ii), then $p_1 \cap p_2 \succeq p'_1 \cap p'_2$ by applying Lemma 19 (ix) twice. For the reverse direction, from Lemma 19 (iii), $p'_1 = (p'_1 \cap p'_2) * z_1$ and $p'_2 = (p'_1 \cap p'_2) * z_2$, for some label-sequences z_1 and z_2. Thus from (iv) and Lemma 19 (iv), $z_1 \neq \epsilon$ and $z_2 \neq \epsilon$ and so $p'_1 \cap p'_2 \succ p'_1$ and $p'_1 \cap p'_2 \succ p'_2$. However by (ii), $p'_1 = p_1 * l_1$ and $p'_2 = p_2 * l_2$, where l_1 and l_2 are label-sequences containing a single label. So since $p'_1 \cap p'_2 \succ p'_1$, we deduce that $p'_1 \cap p'_2 \succeq p_1$ and $p'_1 \cap p'_2 \succeq p_2$ and hence $p'_1 \cap p'_2 \succeq p_1 \cap p_2$ from Lemma 19 (ii). Thus (2) holds.

Suppose first that $closest(v_1, v_2)$ is true, and so there exists $x^1_2 \in [\![p_1 \cap p_2]\!]$ such that $x^1_2 \in aancestor(v_1)$ and $x^1_2 \in ancestor(v_2)$. So by (2), $x^1_2 \in [\![p'_1 \cap p'_2]\!]$ and $x^1_2 \in aancestor(v'_1)$ and $x^1_2 \in aancestor(v'_2)$ from (ii) of the lemma and Lemma 20 (ix). Hence $closest(v'_1, v'_2)$ is true as required.

If instead $closest(v'_1, v'_2)$ is true, then there exists $x'^1_2 \in [\![p'_1 \cap p'_2]\!]$ such that $x'^1_2 \in aancestor(v'_1)$ and $x'^1_2 \in ancestor(v'_2)$. So by (2), $x'^1_2 \in [\![p_1 \cap p_2]\!]$ and since x'^1_2 and v_i are *aancestors* of v'_i for $i = 1, 2$, then $x'^1_2 \in aancestor(v_i)$ for $i = 1, 2$. Hence $closest(v_1, v_2)$ is true and so $closest(v_1, v_2) \iff closest(v'_1, v'_2)$ as claimed.

We next claim that the following holds:

$$p_1 \cap p'_2 = p'_1 \cap p'_2. \tag{3}$$

To show this, since $p_1 \succ p'_1$ from (ii), then by Lemma 19 (ix), $p_1 \cap p'_2 \succeq p'_1 \cap p'_2$. For the reverse direction, from Lemma 19 (iii), $p'_1 = (p'_1 \cap p'_2) * z_1$, for some label-sequence z_1. Thus $z_1 \neq \epsilon$ since otherwise Lemma 19 (iv) would imply that $p'_1 \succeq p'_2$ which contradicts (iii) and (iv), and so $p'_1 \cap p'_2 \succ p'_1$. However by (ii) of the lemma, $p'_1 = p_1 * l_1$, where l_1 is a path sequence containing a single label. So since $p'_1 \cap p'_2 \succ p'_1$ and $p'_1 = p_1 * l_1$, we deduce that $p'_1 \cap p'_2 \succeq p_1$. Also, $p'_1 \cap p'_2 \succeq p'_2$ from Lemma 19 (vii) and so combining this with $p'_1 \cap p'_2 \succeq p_1$ and applying Lemma 19 (ii) shows that $p'_1 \cap p'_2 \succeq p_1 \cap p'_2$, and thus (3) holds.

Suppose then that $closest(v_1, v'_2)$ is true, and so there exists $x^1_2 \in [\![p_1 \cap p'_2]\!]$ such that $x^1_2 \in aancestor(v_1)$ and $x^1_2 \in ancestor(v'_2)$. So by (3), $x^1_2 \in [\![p'_1 \cap p'_2]\!]$ and $x^1_2 \in aancestor(v'_1)$ from (ii) and Lemma 20 (ix). Hence $closest(v'_1, v'_2)$ is true.

If instead $closest(v'_1, v'_2)$ is true, then there exists $x^1_2 \in [\![p'_1 \cap p'_2]\!]$ such that $x^1_2 \in aancestor(v'_1)$ and $x^1_2 \in ancestor(v'_2)$. So by (3), $x^1_2 \in [\![p_1 \cap p'_2]\!]$. It follows from Lemma 19 (vii) that $p_1 \cap p'_2 \succeq p_1$, and combining this with the fact that both v_1 and x^1_2 are *aancestors* of v'_1 and using Lemma 20 (viii) shows that $x^1_2 \in aancestor(v_1)$, and so $closest(v_1, v'_2)$ is true and hence $closest(v_1, v'_2) \iff closest(v'_1, v'_2)$ as claimed, which completes the proof. □

Using this lemma, we now establish the important preliminary result which shows that an XML tree $\mathbb{T}$ satisfies an XFD if and only if $\alpha(\mathbb{T})$ satisfies the corresponding text XFD.

Theorem 30. *Let $\mathbb{P}$ be a set of paths, σ an XFD and $\mathbb{T}$ an XML tree, where $\mathbb{T} \models \mathbb{P}$ and $\sigma \models \mathbb{P}$. Then $\mathbb{T} \models \sigma$ iff $\alpha(\mathbb{T}) \models \alpha(\sigma)$.*

Proof. Let σ be the XFD $p_1, \ldots, p_n \to p_{n+1}$.

If: The argument is by contra-positive, and so we will show that if $\mathbb{T} \not\models \sigma$ then $\alpha(\mathbb{T}) \not\models \alpha(\sigma)$. By definition of $\alpha(\sigma)$, one of the following cases must occur in σ:

(a) $\forall i \in [1, n+1]$, $p_i \notin ele(\mathbb{P})$ (every path in σ is a text path);
(b) $p_{n+1} \in ele(\mathbb{P})$ and $\forall i \in [1, n]$, $p_i \notin ele(\mathbb{P})$(every path on the l.h.s. of σ is a text path but the r.h.s. path is not);
(c) $\exists j \in [1, n]$ such that $\forall i \in [1, j]$, $p_i \in ele(\mathbb{P})$ and $\forall i \in [j+1, n+1]$, $p_i \notin ele(\mathbb{P})$ (permuting subscripts might be necessary) (at least one path on the l.h.s. of σ is not a text path and the r.h.s. path is a text path);
(d) $\exists j \in [1, n]$ such that $\forall i \in [1, j]$, $p_i \in ele(\mathbb{P})$ and $\forall i \in [j+1, n]$, $p_i \notin ele(\mathbb{P})$ and $p_{n+1} \in ele(\mathbb{P})$ (permuting subscripts might be necessary)(at least one path on the l.h.s. of σ is not a text path and the r.h.s. path is not a text path).

We first note that it follows directly from the definition of α that $\mathbb{T} \sqsubseteq \alpha(\mathbb{T})$.

For case (a), by definition of α, $\alpha(\sigma) = \sigma$, and so by Lemma 28 and the fact that $\mathbb{T} \sqsubseteq \alpha(\mathbb{T})$, if $\mathbb{T} \not\models \sigma$ then $\alpha(\mathbb{T}) \not\models \alpha(\sigma)$ as required.

For case (b), $\alpha(\sigma) = p_1, \ldots, p_n \to p_{n+1} * l_@$. Since $\mathbb{T} \not\models \sigma$, then (iii) of Definition 22 holds (with $m = 1$) and by definition of $\alpha(\mathbb{T})$, there exist a node $\bar{v}_{n+1} \in [\![p_{n+1} * l_@]\!]$ in $\alpha(\mathbb{T})$ such that $v_{n+1} = parent(\bar{v}_{n+1})$, and a node $\bar{v}'_{n+1} \in [\![p_{n+1} * l_@]\!]$ in $\alpha(\mathbb{T})$ such that $v'_{n+1} = parent(\bar{v}'_{n+1})$. Consider then the nodes $v_1, \ldots, v_n, \bar{v}_{n+1}$ in $\alpha(\mathbb{T})$. We claim that the nodes in this set pairwise satisfy the *closest* property. To verify this, if both nodes are in $v_1, \ldots, v_n$ then it follows from the definition of $\alpha(\mathbb{T})$ that both nodes are also in $\mathbb{T}$ and thus the property follows from (iii) of Definition 22. If one node is in $v_1, \ldots, v_n$ and the other is $\bar{v}_{n+1}$, then it follows from (iii) of Definition 22 and Lemma 29 that the nodes satisfy the *closest* property. Using the same reasoning, the pairwise *closest* property also holds for the set of nodes $v'_1, \ldots, v'_n, \bar{v}_{n+1}$. Then since $val(v_{n+1}) \neq val(v'_{n+1})$ from (iii) of Definition 22, $v_{n+1} \neq v'_{n+1}$ and so $\bar{v}_{n+1} \neq \bar{v}'_{n+1}$ and thus $val(\bar{v}_{n+1}) \neq val(\bar{v}'_{n+1})$ since both nodes have element labels from (b). Combining this with (iii) of Definition 22 shows that $\alpha(\mathbb{T}) \not\models \alpha(\sigma)$ as required.

Consider case (c). In this case, $\alpha(\sigma) = p_1 * l_@, \ldots, p_j * l_@, p_{j+1}, \ldots, p_n \to p_{n+1}$. Since $\mathbb{T} \not\models \sigma$, (iii) of Definition 22 holds and because paths $p_1, \ldots, p_j$ end in labels in $\mathbf{E}$ by (c), it follows from the definition of *val* and (iii) of Definition 22 that

$$\forall i \in [1, j], \quad v_i = v'_i. \tag{4}$$

Consider the paths $p_1, \ldots, p_j$. By the definition of $\alpha(\mathbb{T})$ and (c), corresponding to $v_1, \ldots, v_j$ there exist nodes $\bar{v}_1 \in [\![p_1 * l_@]\!], \ldots, \bar{v}_j \in [\![p_j * l_@]\!]$ in $\alpha(\mathbb{T})$ such that $parent(\bar{v}_1) = v_1, \ldots, parent(\bar{v}_j) = v_j$, and corresponding to $v'_1, \ldots, v'_j$ there exist nodes $\bar{v}'_1 \in [\![p_1 * l_@]\!], \ldots, \bar{v}'_j \in [\![p_j * l_@]\!]$ in $\alpha(\mathbb{T})$ such that $parent(\bar{v}'_1) = v'_1, \ldots, parent(\bar{v}'_j) = v'_j$. Consider then the set of nodes $\bar{v}_1, \ldots, \bar{v}_j, v_{j+1}, \ldots, v_{n+1}$ in $\alpha(\mathbb{T})$. We note that $closest(\bar{v}_i, \bar{v}_k)$ is true for all $i, k \in [1, j]$ by (iii) of Definition 22 and Lemma 29, $closest(\bar{v}_i, v_k)$ is true for $i \in [1, j]$ and $k \in [j+1, n+1]$ by (iii) of Definition 22 and Lemma 29, and $closest(v_i, v_k)$ is true for $i, k \in [j+1, n+1]$ from (iii) of Definition 22. So the following holds in $\alpha(\mathbb{T})$:

$$\forall v, v' \in \{\bar{v}_1, \ldots, \bar{v}_j, v_{j+1}, \ldots, v_{n+1}\}, \quad closest(v, v') \text{ is true.} \tag{5}$$

Using the same arguments, for the set of nodes $\bar{v}'_1, \ldots, \bar{v}'_j, v'_{j+1}, \ldots, v'_{n+1}$ in $\alpha(\mathbb{T})$, where $\bar{v}'_1 = parent(v'_1), \ldots, \bar{v}'_j = parent(v_j)$, the following holds in $\alpha(\mathbb{T})$:

$$\forall v, v' \in \{\bar{v}'_1, \ldots, \bar{v}'_j, v'_{j+1}, \ldots, v'_{n+1}\}, \quad closest(v, v') \text{ is true.} \tag{6}$$

Then, it follows from (4) and the fact that all the paths $p_1 * l_@, \ldots, p_j * l_@$ end in an attribute label that corresponding nodes in $\bar{v}_1, \ldots, \bar{v}_j$ and $\bar{v}'_1, \ldots, \bar{v}'_j$ are identical, and thus have the same *val*, and combining this with (iii)-(b) and (iii)-(c) of Definition 22 shows that $\alpha(\mathbb{T}) \not\models \alpha(\sigma)$ as required.

Consider (d). In this case $\alpha(\sigma) = p_1 * l_@, \ldots, p_j * l_@, p_{j+1}, \ldots, p_n \to p_{n+1} * l_@$. Once again, because $\mathbb{T} \not\models \sigma$ then (iii) of Definition 22 holds and because of (d), (4) holds. Consider the paths $p_1, \ldots, p_j, p_{n+1}$. By the definition of $\alpha(\mathbb{T})$, there exist nodes $\bar{v}_1 \in [\![p_1 * l_@]\!], \ldots, \bar{v}_j \in [\![p_j * l_@]\!], \bar{v}_{n+1} \in [\![p_{n+1} * l_@]\!]$ in $\alpha(\mathbb{T})$ such that $parent(\bar{v}_1) = v_1, \ldots, parent(\bar{v}_j) = v_j, parent(\bar{v}_{n+1}) = v_{n+1}$, and nodes $\bar{v}'_1 \in [\![p_1 * l_@]\!], \ldots, \bar{v}'_j \in [\![p_j * l_@]\!], \bar{v}'_{n+1} \in [\![p_{n+1} * l_@]\!]$ in $\alpha(\mathbb{T})$ such that $parent(\bar{v}'_1) = v'_1, \ldots, parent(\bar{v}'_j) = v'_j, parent(\bar{v}'_{n+1}) = v'_{n+1}$. Consider then the nodes $v'_1, \ldots, v'_j, v_{j+1}, \ldots, v_n, v'_{n+1}$. Using the same arguments as in (c), nodes in this set pairwise satisfy the *closest* property as do nodes in the set $\bar{v}'_1, \ldots, \bar{v}'_j, v_{j+1}, \ldots, v_n, \bar{v}'_{n+1}$. Since $Last(p_{n+1}) \in \mathbf{E}$ from (d) and $val(v_{n+1}) \neq val(v'_{n+1})$ from (iii) of Definition 22, then $v_{n+1} \neq v'_{n+1}$ and so $val(\bar{v}_{n+1}) \neq val(\bar{v}'_{n+1})$ by definition of $\alpha(\mathbb{T})$. Combining this with (iii) of Definition 22 shows that $\alpha(\mathbb{T}) \not\models \alpha(\sigma)$ as required. This completes the proof of the *If* part.

Only If: The proof is by contra-positive, and so we show that if $\alpha(\mathbb{T}) \not\models \alpha(\sigma)$ then $\mathbb{T} \not\models \sigma$. Since $\alpha(\mathbb{T}) \not\models \alpha(p_1), \ldots, \alpha(p_n) \to \alpha(p_{n+1})$, there exist nodes $\{v_1, v'_1\} \subseteq [\![\alpha(p_1)]\!], \ldots, \{v_{n+1}, v'_{n+1}\} \subseteq [\![\alpha(p_{n+1})]\!]$ in $\alpha(\mathbb{T})$ such that (iii) of Definition 22 holds.

As for the *If* part, one of the cases (a)–(d) must hold for σ.

Consider case (a). By definition of $\alpha(p)$ and (a), $\alpha(\sigma) = \sigma$ and so $\{v_1, v'_1\}, \ldots, \{v_{n+1}, v'_{n+1}\}$ are also in $\mathbb{T}$ by definition of $\alpha(\mathbb{T})$. So since (iii) of Definition 22 holds for $\alpha(\mathbb{T})$, it must also hold for $\mathbb{T}$ and so $\mathbb{T} \not\models \sigma$ as required.

Consider (b). Since $p_{n+1} \in ele(\mathbb{P})$ from (b), $\alpha(p_{n+1}) = p_{n+1} * l_@$ and so $Parent(\alpha(p_{n+1})) = p_{n+1}$. Let $\bar{v}_{n+1}$ be the node in $[\![p_{n+1}]\!]$ in $\alpha(\mathbb{T})$ such that $parent(v_{n+1}) = \bar{v}_{n+1}$, and let $\bar{v}'_{n+1}$ be the node in $[\![p_{n+1}]\!]$ in $\alpha(\mathbb{T})$ such that $parent(v'_{n+1}) = \bar{v}'_{n+1}$. Consider the nodes $v_1, \ldots, v_n, \bar{v}_{n+1}$ and $v'_1, \ldots, v'_n, \bar{v}'_{n+1}$ in $\alpha(\mathbb{T})$. From (b), the final label in each of the paths $p_1, \ldots, p_n$ is not in $\mathbf{E}$, so by definition of $\alpha(\mathbb{T})$ the nodes $v_1, \ldots, v_n$ and $v'_1, \ldots, v'_n$ are also in $\mathbb{T}$, as also are $\bar{v}_{n+1}$ and $\bar{v}'_{n+1}$. Also, for every $i, j \in [1, n]$, $closest(v_i, v_j)$ is true in $\mathbb{T}$ by (iii) of Definition 22, and for all $i \in [1, n]$, $closest(v_i, \bar{v}_{n+1})$ is true by (iii) of Definition 22 and Lemma 29. Using the same reasoning, for all $i, j \in [1, n]$, $closest(v'_i, v'_j)$ is true in $\mathbb{T}$ and $closest(v'_i, \bar{v}'_{n+1})$

is true. Also, since $v_{n+1} \neq v'_{n+1}$ by (iii) of Definition 22 and the path $\alpha(p_{n+1})$ by definition ends in the attribute label $l_@$, then $\bar{v}_{n+1} \neq \bar{v}'_{n+1}$ and since $lab(\bar{v}_{n+1}) \in \mathbf{E}$, $val(\bar{v}_{n+1}) \neq val(\bar{v}'_{n+1})$. Combining this with (iii) of Definition 22 then shows that $\mathbb{T} \not\models \sigma$, which was to be established.

Consider next case (c). Since $p_i \in ele(\mathbb{P})$ for all $i \in [1, j]$ from (c), then $\alpha(p_i) = p_i * l_@$ and so $Parent(\alpha(p_i)) = p_i$. Consider then the set of nodes $\bar{v}_1, \ldots, \bar{v}_j, v_{j+1}, \ldots, v_{n+1}$ in $\alpha(\mathbb{T})$, where $\bar{v}_1 = parent(v_1), \ldots, \bar{v}_j = parent(v_j)$. From the definition of $\alpha(\mathbb{T})$, $\bar{v}_1, \ldots, \bar{v}_j, v_{j+1}, \ldots, v_{n+1}$ are also in $\mathbb{T}$. We note that for all $i, k \in [1, j]$, $closest(\bar{v}_i, \bar{v}_k)$ is true by (iii) of Definition 22 and Lemma 29; $closest(\bar{v}_i, v_k)$ is true for all $i \in [1, j]$ and $k \in [j+1, n+1]$ by (iii) of Definition 22 and Lemma 29; and $closest(v_i, v_k)$ is true for all $i, k \in [j+1, n+1]$ from (iii) of Definition 22. So the following holds:

$$\forall v, v' \in \{\bar{v}_1, \ldots, \bar{v}_j, v_{j+1}, \ldots, v_{n+1}\}, \quad closest\big(v, v'\big) \text{ is true.} \tag{7}$$

Using the same arguments, for the set of nodes $\bar{v}'_1, \ldots, \bar{v}'_j, v'_{j+1}, \ldots, v'_{n+1}$ in $\alpha(\mathbb{T})$, where $\bar{v}'_1 = parent(v'_1), \ldots, \bar{v}'_j = parent(v_j)$, the following holds:

$$\forall v, v' \in \big\{\bar{v}'_1, \ldots, \bar{v}'_j, v'_{j+1}, \ldots, v'_{n+1}\big\}, \quad closest\big(v, v'\big) \text{ is true.} \tag{8}$$

Consider then any pair of nodes v_i, v'_i, where $i \in [1, j]$. Since $val(v_i) = val(v'_i)$ from (iii) of Definition 22, we must have that $v_i = v'_i$ since $\mathbb{T} \models \alpha(\Sigma)$ and thus $\mathbb{T} \models \alpha(p_i) \to p_i$. So $\bar{v}_i = \bar{v}'_i$ and also thus $val(\bar{v}_i) = val(\bar{v}'_i)$. Combining this with (iii) of Definition 22, it follows that the val of every node in the set $\bar{v}_1, \ldots, \bar{v}_j, v_{j+1}, \ldots, v_n$ is equal to the val of the corresponding node in the set $\bar{v}'_1, \ldots, \bar{v}'_j, v'_{j+1}, \ldots, v'_n$, and $val(v_{n+1}) \neq val(v'_{n+1})$ from (iii) of Definition 22. It follows from this and (8) and (7) that $\alpha(\mathbb{T}) \not\models p_1, \ldots, p_j, p_{j+1}, \ldots, p_n \to p_{n+1}$, and hence $\mathbb{T} \not\models p_1, \ldots, p_j, p_{j+1}, \ldots, p_n \to p_{n+1}$ since, by definition of $\alpha(\mathbb{T})$, the sets of nodes $\bar{v}_1, \ldots, \bar{v}_j, v_{j+1}, \ldots, v_{n+1}$ and $\bar{v}'_1, \ldots, \bar{v}'_j, v'_{j+1}, \ldots, v'_{n+1}$ are also in $\mathbb{T}$. This completes (c).

Lastly, consider (d) and the sets of nodes $\bar{v}_1, \ldots, \bar{v}_j, v_{j+1}, \ldots, v_n, \bar{v}_{n+1}$, where $\bar{v}_1 = parent(v_1), \ldots, \bar{v}_j = parent(v_j)$, $\bar{v}_{n+1} = parent(v_{n+1})$; and $\bar{v}'_1, \ldots, \bar{v}'_j, v'_{j+1}, \ldots, v'_n, v'_{n+1}$ in $\mathbb{T}$, where $\bar{v}'_1 = parent(v'_1), \ldots, \bar{v}'_j = parent(v_j), \bar{v}'_{n+1} = parent(v_{n+1})$. We firstly note that since $val(v_{n+1}) \neq val(v'_{n+1})$ from (iii) of Definition 22, and $lab(v_{n+1}) = lab(v'_{n+1}) = l_@$ since $\alpha(p_{n+1}) = p_{n+1} * l_@$ from (d), then $\bar{v}_{n+1} \neq \bar{v}_{n+1}$ and so $val(\bar{v}_{n+1}) \neq val(\bar{v}_{n+1})$ since the label of both nodes is in $\mathbf{E}$. It then follows, using the same arguments in (c), that $\alpha(\mathbb{T}) \not\models \alpha(p_1), \ldots, \alpha(p_j), p_{j+1}, \ldots, p_n \to \alpha(p_{n+1})$, and so $\mathbb{T} \not\models \sigma$, which completes (d) and the proof. □

This leads to the first main result of this section, which shows that a set of XFDs Σ implies an XFD σ if and only if the corresponding set of text XFDs $\alpha(\Sigma)$ implies the corresponding text XFD $\alpha(\sigma)$.

Theorem 31. *Let $\mathbb{P}$ be a set of paths, Σ a set of XFDs and σ an XFD such that $\Sigma \models \mathbb{P}$ and $\sigma \models \mathbb{P}$. Then $(\mathbb{P}, \Sigma) \vdash \sigma$ iff $(\alpha(\mathbb{P}), \alpha(\Sigma)) \vdash \alpha(\sigma)$.*

Proof. *Only If*: The proof is by contra-positive, and so we show that if $(\alpha(\mathbb{P}), \alpha(\Sigma)) \nvdash \alpha(\sigma)$ then $(\mathbb{P}, \Sigma) \nvdash \sigma$. If $(\alpha(\mathbb{P}), \alpha(\Sigma)) \nvdash \alpha(\sigma)$, there exists a complete tree $\mathbb{T}'$ such that $\mathbb{T}' \models \alpha(\mathbb{P})$ and $\mathbb{T}' \models \alpha(\Sigma)$, but $\mathbb{T}' \not\models \alpha(\sigma)$.

Since $\mathbb{T}' \models \alpha(\Sigma)$ and by definition of $\alpha(\Sigma)$, $\mathbb{T}' \models \alpha(p) \to p$ for every $p \in ele(\mathbb{P})$, then the val of every node with label $l_@$ in $\mathbb{T}'$ must be distinct within the set of nodes $[\![\alpha(p)]\!]$ (but not necessarily in the whole of $\mathbb{T}'$). So from $\mathbb{T}'$ construct a new tree $\mathbb{T}''$ by replacing the val of every node labeled with $l_@$ with a new val so that the new val is unique in the whole tree $\mathbb{T}''$, rather than just being unique within each $[\![\alpha(p)]\!]$. We next claim that given an arbitrary XFD σ_1 such that $\sigma_1 \models \mathbb{P}$, then $\mathbb{T}' \models \alpha(\sigma_1)$ iff $\mathbb{T}'' \models \alpha(\sigma_1)$. We first show that if $\mathbb{T}' \not\models \alpha(\sigma_1)$ then $\mathbb{T}'' \not\models \alpha(\sigma_1)$. To verify this, let $\alpha(\sigma_1) = \alpha(p_1), \ldots, \alpha(p_n) \to \alpha(p_{n+1})$ and so the conditions of Definition 22 are satisfied. Now for any $v_i \in \{v_1, \ldots, v_n\}$ and $v'_i \in \{v'_1, \ldots, v'_n\}$, if $lab(v_i) = lab(v'_i) \neq l_@$ then $val(v_i)$ and $val(v'_i)$ are the same in both $\mathbb{T}'$ and $\mathbb{T}''$ and so if $val(v_i) = val(v'_i)$ in $\mathbb{T}'$ from Definition 22 then $val(v_i) = val(v'_i)$ in $\mathbb{T}''$. Similarly, if $lab(v_{n+1}) = lab(v'_{n+1}) \neq l_@$ and $val(v_{n+1}) \neq val(v'_{n+1})$ in $\mathbb{T}'$ then $val(v_{n+1}) \neq val(v'_{n+1})$ in $\mathbb{T}''$ since $val(v_{n+1})$ and $val(v'_{n+1})$ are also unchanged. Alternatively, if $lab(v_i) = lab(v'_i) = l_@$, then $\{v_i, v'_i\} \subseteq [\![p_i * l_@]\!]$. So since the val's of nodes in $[\![p_i * l_@]\!]$ are unique in $\mathbb{T}'$ and $val(v_i) = val(v'_i)$ from Definition 22, this means $v_i = v'_i$ and so even if $val(v_i)$ is changed, then $val(v_i) = val(v'_i)$ also in $\mathbb{T}''$ also. Similarly, if $lab(v_{n+1}) = lab(v'_{n+1}) = l_@$ and $val(v_{n+1}) \neq val(v'_{n+1})$ in $\mathbb{T}'$ then $val(v_{n+1}) \neq val(v'_{n+1})$ in $\mathbb{T}''$ since the val of every node in $\mathbb{T}''$ with label $l_@$ is unique. So this means that $\mathbb{T}'' \not\models \alpha(\sigma_1)$ as claimed. Conversely, to show that if $\mathbb{T}'' \not\models \alpha(\sigma_1)$ then $\mathbb{T}' \not\models \alpha(\sigma_1)$ consider first from Definition 22 any $v_i \in \{v_1, \ldots, v_n\}$ and $v'_i \in \{v'_1, \ldots, v'_n\}$ in $\mathbb{T}''$. Then the same arguments just used show that if $val(v_i) = val(v'_i)$ in $\mathbb{T}''$ then $val(v_i) = val(v'_i)$ in $\mathbb{T}'$. Similarly, if $val(v_{n+1}) \neq val(v'_{n+1})$ in $\mathbb{T}''$ and $lab(v_{n+1}) = lab(v'_{n+1}) \neq l_@$, then $val(v_{n+1})$ and $val(v'_{n+1})$ have not been changed and so $val(v_{n+1}) \neq val(v'_{n+1})$ in $\mathbb{T}'$. Alternatively, if $lab(v_{n+1}) = lab(v'_{n+1}) = l_@$ then since v_{n+1} and v'_{n+1} are both in $[\![p_{n+1} * l_@]\!]$, then $val(v_{n+1}) \neq val(v_{n+1})$ in $\mathbb{T}'$ since the val of nodes in $[\![p_{n+1} * l_@]\!]$ are distinct in $\mathbb{T}'$. This means that if $\mathbb{T}'' \not\models \alpha(\sigma_1)$ then $\mathbb{T}' \not\models \alpha(\sigma_1)$, which establishes our claim. As a consequence, $\mathbb{T}' \models \alpha(\Sigma)$ and $\mathbb{T}' \not\models \alpha(\sigma)$ iff $\mathbb{T}'' \models \alpha(\Sigma)$ and $\mathbb{T}'' \not\models \alpha(\sigma)$.

From $\mathbb{T}''$ construct a tree $\mathbb{T}$ by removing all nodes labeled with $l_@$. It is easy to verify that $\mathbb{T}$ is complete and conforms to $\mathbb{P}$. From this it follows that $\mathbb{T}'' = \alpha(\mathbb{T})$ since the val of every node in $\mathbb{T}''$ with label $l_@$ is unique. Thus since $\alpha(\mathbb{T}) = \mathbb{T}'' \models \alpha(\Sigma)$ and $\alpha(\mathbb{T}) \not\models \alpha(\sigma)$, it follows from Theorem 30 that $\mathbb{T} \models \Sigma$ and $\mathbb{T} \not\models \sigma$, so $(\mathbb{P}, \Sigma) \nvdash \sigma$.

If: The proof is by contra-positive, and so we show that if $(\mathbb{P}, \Sigma) \nvdash \sigma$ then $(\alpha(\mathbb{P}), \alpha(\Sigma)) \nvdash \alpha(\sigma)$. If $(\mathbb{P}, \Sigma) \nvdash \sigma$, there exists a complete tree $\mathbb{T}$ such that $\mathbb{T} \models \mathbb{P}$, $\mathbb{T} \models \Sigma$, but $\mathbb{T} \not\models \sigma$. As noted earlier, $\alpha(\mathbb{T})$ is complete and conforms to $\alpha(\mathbb{P})$. It follows then from Theorem 30 that $\alpha(\mathbb{T}) \models \alpha(\Sigma)$ but $\alpha(\mathbb{T}) \not\models \alpha(\sigma)$, and so $(\alpha(\mathbb{P}), \alpha(\Sigma)) \nvdash \alpha(\sigma)$. □

The next important property of a set of text XFDs $\alpha(\Sigma)$ is that any derivation sequence involving the XFD axioms and $\alpha(\Sigma)$ can be converted to a derivation involving Σ. To establish this result, we first prove a preliminary lemma.

Lemma 32. *If $\mathbb{P}$ is a set of paths and $\{p, q, r\} \subseteq \mathbb{P}$, and $\alpha(p) \in \alpha(\mathbb{P})$ and $\alpha(q) \in \alpha(\mathbb{P})$, then*:

(i) $\alpha(p) \succeq \alpha(q) \Rightarrow p \succeq q$;
(ii) $\alpha(p) \cap q = p \cap \alpha(q)$;
(iii) *if $p \neq q$ then $\alpha(p) \cap \alpha(q) = p \cap q$;*
(iv) *if $\alpha(p) \cap \alpha(q) \succeq \alpha(r)$, then $p \cap q \succeq r$.*

Proof. We establish (i) and (ii) first. The results are immediate if $p = q$ so we assume that $p \neq q$. We then consider the four possible cases:

(a) $\alpha(p) = p$ and $\alpha(q) = q$;
(b) $\alpha(p) = p$ and $q \succ \alpha(q)$;
(c) $p \succ \alpha(p)$ and $\alpha(q) = q$;
(d) $p \succ \alpha(p)$ and $q \succ \alpha(q)$.

For (a), both (i) and (ii) follow immediately.

If (b) holds, then since $q \succ \alpha(q)$ it follows from the definition of α that $\alpha(q) = q * l_@$, and so $\alpha(p) \succeq \alpha(q)$ becomes $p \succeq (q * l_@)$. However if $\alpha(p) = p$ and $\alpha(q) = q * l_@$, then by definition of α, p ends in a non-element label and q ends in an element label and so if $p \succeq (q * l_@)$ then it follows from the definition of $\succeq$ that $p \succeq q$ and so (i) holds. For (ii), $\alpha(p) \cap q = p \cap q$ and $p \cap \alpha(q) = p \cap (q * l_@)$. However since p ends in a text or attribute label which is not equal to $l_@$ by definition of α, then $p \cap (q * l_@) = p \cap q$ and so (ii) holds.

If (c) holds, then by symmetry the same argument as in (b) applies.

If (d) holds, from the definition of α we have that $\alpha(p) = p * l_@$ and $\alpha(q) = q * l_@$, and so $\alpha(p) \succeq \alpha(q)$ becomes $p * l_@ \succeq q * l_@$. Clearly, if this condition holds then $p \succeq q$ and so (i) holds. For (ii), $\alpha(p) \cap q = (p * l_@) \cap q$ which is equal to $p \cap q$ since the last label of q cannot be $l_@$. By symmetry, the same argument applies to $p \cap \alpha(q)$ and thus (ii) holds.

This establishes (i) and (ii), so we now consider (iii) and (iv). For (iii), one of the cases (a)–(d) must occur. If (a) holds, then it is immediate that $\alpha(p) \cap \alpha(q) = p \cap q$. If (b) occurs, then from the definition of α we have that $\alpha(p) = p$ and $\alpha(q) = q * l_@$, and so $\alpha(p) \cap \alpha(q) = p \cap (q * l_@)$. However, by definition of α, p cannot end with label $l_@$ and so $p \cap (q * l_@) = p \cap q$ and so (iii) holds. If (c) holds, from the definition of α we have that $\alpha(p) = p * l_@$ and $\alpha(q) = q$, and so $\alpha(p) \cap \alpha(q) = (p * l_@) \cap q$ and $(p * l_@) \cap q = p \cap q$ since q cannot end in with the label $l_@$ and so (iii) holds. If (d) holds, from the definition of α we have that $\alpha(p) = p * l_@$ and $\alpha(q) = q * l_@$, and so $\alpha(p) \cap \alpha(q)$ becomes $(p * l_@) \cap (q * l_@)$. However since $p \neq q$, then $(p * l_@) \cap (q * l_@) = p \cap q$ from the definition of $\cap$. This completes (iii).

For (iv), if $p = q$ then $\alpha(p) \cap \alpha(q) = \alpha(p)$ and so if $\alpha(p) \cap \alpha(q) \succeq \alpha(r)$ then $\alpha(p) \succeq \alpha(r)$ and using (i) shows that $p \succeq r$ as required. If instead $p \neq q$, then by (iii), (iv) becomes $p \cap q \succeq \alpha(r)$. However, by definition of α, neither p nor q can end with the label $l_@$ and so if $p \cap q \succeq \alpha(r)$, then $p \cap q \succeq r$ as required. □

We now establish the second main result of the section, which shows that any derivation of a text XFD from a set of text XFDs can be converted to a derivation of the original XFD from the original set of XFDs.

Theorem 33. *If $\mathbb{P}$ is a set of paths and Σ a set of XFDs such that $\Sigma \models \mathbb{P}$ and σ an XFD such that $\sigma \models \mathbb{P}$ and $(\alpha(\mathbb{P}), \alpha(\Sigma)) \vdash^d \alpha(\sigma)$, then $(\mathbb{P}, \Sigma) \vdash^d \sigma$.*

Proof. We first note that it follows directly from the definitions of α that if $\Sigma \models \mathbb{P}$ and $\sigma \models \mathbb{P}$, then $\alpha(\Sigma) \models \alpha(\mathbb{P})$ and $\alpha(\sigma) \models \alpha(\mathbb{P})$. We also note that for every XFD σ such that $\sigma \models \alpha(\mathbb{P})$, $\sigma = \alpha(\sigma')$ for some σ' such that $\sigma' \models \mathbb{P}$. We prove the result by induction on the number of XFDs in the derivation $\alpha(\sigma_1), \ldots, \alpha(\sigma_m) = \alpha(\sigma)$. Initially, the result is true when $m = 1$ since, by definition of $\alpha(\Sigma)$, if $\alpha(\sigma) \in \alpha(\Sigma)$ then $\sigma \in \Sigma$. Suppose then inductively the result holds for a derivation sequence of $m - 1$ XFDs. Without any loss of generality, by Axioms A1–A3 we can assume that there is a single path on the r.h.s. of σ, and so let σ be the XFD $p_1, \ldots, p_n \rightarrow p_{n+1}$ and thus $\alpha(\sigma) = \alpha(p_1), \ldots, \alpha(p_n) \rightarrow \alpha(p_{n+1})$.

Let τ be the axiom used to derive σ. If $\tau = A1$, then $\alpha(p_{n+1}) = \alpha(p_i)$, for some $i \in [1, n]$, and so $p_{n+1} = p_i$ and thus $(\mathbb{P}, \Sigma) \vdash^d \sigma$ by Axiom A1.

In the case $\tau = A2$, we can assume without loss of generality that $\alpha(\sigma_{m-1}) = \alpha(p_2), \ldots, \alpha(p_n) \rightarrow \alpha(p_{n+1})$ and so $(\mathbb{P}, \Sigma) \vdash^d \sigma_{m-1} = p_2, \ldots, p_n \rightarrow p_{n+1}$ from the inductive assumption and hence $(\mathbb{P}, \Sigma) \vdash^d \sigma$ from Axiom A2.

If $\tau = A3$, there exist $\alpha(\sigma_j) = \alpha(p_1), \ldots, \alpha(p_n) \rightarrow \alpha(q_1), \ldots, \alpha(q_r)$ and $\alpha(\sigma_k) = \alpha(q_1), \ldots, \alpha(q_r) \rightarrow \alpha(p_{n+1})$, where $j, k \in [1, m-1]$ and $(\alpha(\mathbb{P}), \alpha(\Sigma)) \vdash^d \alpha(\sigma_j)$ and $(\alpha(\mathbb{P}), \alpha(\Sigma)) \vdash^d \alpha(\sigma_k)$. By the inductive hypothesis, $(\mathbb{P}, \Sigma) \vdash^d p_1, \ldots, p_n \rightarrow q_1, \ldots, q_r$ and $(\mathbb{P}, \Sigma) \vdash^d q_1, \ldots, q_r \rightarrow p_{n+1}$, and so $(\mathbb{P}, \Sigma) \vdash^d p_1, \ldots, p_n \rightarrow p_{n+1}$ by Axiom A3.

If $\tau = A4$, there exists $\alpha(\sigma_j) = \alpha(\bar{p}_1), \ldots, \alpha(\bar{p}_r) \rightarrow \alpha(p_{n+1})$, where $j \in [1, m-1]$, such that $(\alpha(\mathbb{P}), \alpha(\Sigma)) \vdash^d \alpha(\sigma_j)$ and for all $i \in [1, r], \alpha(\bar{p}_i) \cap \alpha(p_{n+1}) = \rho$. So by the inductive hypothesis, $(\mathbb{P}, \Sigma) \vdash^d \bar{p}_1, \ldots, \bar{p}_r \rightarrow p_{n+1}$. We can assume that $\forall i \in [1, r], \alpha(\bar{p}_i) \neq \alpha(p_{n+1})$ since otherwise the case reduces to A1. If $\alpha(p_{n+1}) = p_{n+1}$, then for all $i \in [1, r], \alpha(\bar{p}_i) \cap p_{n+1} =$

$\bar{p}_i \cap \alpha(p_{n+1})$ by Lemma 32 (ii) and hence $\alpha(\bar{p}_i) \cap p_{n+1} = \bar{p}_i \cap p_{n+1}$, and then applying the conditions of A4 shows that $\bar{p}_i \cap p_{n+1} = \rho$ and hence $(\mathbb{P}, \Sigma) \models^d \sigma$ follows by Axiom A4. Alternatively, if $\alpha(p_{n+1}) \neq p_{n+1}$, then from Lemma 32 (iii) for all $i \in [1, r]$, $\alpha(\bar{p}_i) \cap \alpha(p_{n+1}) = \bar{p}_i \cap p_{n+1}$ and so $p_i \cap p_{n+1} = \rho$ since by assumption $\alpha(\bar{p}_i) \cap \alpha(p_{n+1}) = \rho$. Hence $(\mathbb{P}, \Sigma) \models^d \sigma$ follows again by Axiom A4.

If $\tau = A5$, there exists $\alpha(\sigma_j) = \alpha(\bar{p}_1), \ldots, \alpha(\bar{p}_n) \to \alpha(p_{n+1})$, such that $(\alpha(\mathbb{P}), \alpha(\Sigma)) \models^d \alpha(\sigma_j)$ and for all $i \in [1, n]$, $\alpha(\bar{p}_i) \cap \alpha(p_{n+1}) \succeq \alpha(p_i)$ and $(\alpha(p_i) \succeq \alpha(\bar{p}_i)$ or $\alpha(p_i) \succeq \alpha(p_{n+1}))$. Since $\alpha(\bar{p}_i) \cap \alpha(p_{n+1}) \succeq \alpha(p_i)$, it follows from Lemma 32 (iv) that $\bar{p}_i \cap p_{n+1} \succeq p_i$. Also, $p_i \succeq \bar{p}_i$ and $p_i \succeq p_{n+1}$ follow from Lemma 32 (i) and so we can apply Axiom A5 to $\bar{p}_1, \ldots, \bar{p}_n \to p_{n+1}$ and hence $(\mathbb{P}, \Sigma) \models^d \sigma$ as required.

Consider $\tau = A6$. If this rule applies, then $\alpha(\sigma) = \alpha(p_1) \to \alpha(p_{n+1})$, $Last(\alpha(p_1)) \in \mathbf{E}$ and $\alpha(p_{n+1}) \succ \alpha(p_1)$. However, by definition of α, the only way that this can occur is if $\alpha(p_1) = p_1$ and $\alpha(p_{n+1}) = p_{n+1}$. In this case, $(\mathbb{P}, \Sigma) \vdash p_1 \to p_{n+1}$ from A6.

Consider $\tau = A7$. If this rule applies, then $\alpha(\sigma) = \alpha(p_1) \to \alpha(p_{n+1})$, where $Last(\alpha(p_{n+1})) \in \mathbf{A}$ and $\alpha(p_1) = Parent(\alpha(p_{n+1}))$. Again, by definition of α, the only way that this can occur is if $\alpha(p_1) = p_1$ and $\alpha(p_{n+1}) = p_{n+1}$. In this case, $(\mathbb{P}, \Sigma) \vdash p_1 \to p_{n+1}$ from A7.

If $\tau = A8$, then $\alpha(p_{n+1}) = \alpha(\rho)$ and so $(\mathbb{P}, \Sigma) \models^d p_1, \ldots, p_n \to \rho$ by Axiom A8. □

4.2. Establishing completeness

In this section we show that the XFD axiom system introduced previously is complete, that is if $(\mathbb{P}, \Sigma) \vdash \sigma$ then $(\mathbb{P}, \Sigma) \models^d \sigma$, and that the implication problem is decidable in quadratic time. We do this in several steps. Firstly, we introduce a closure algorithm and show that it is sound for sets of arbitrary XFDs. Secondly, we show that the closure algorithm is also complete for sets of text XFDs by introducing a chase style algorithm for XML trees. Finally, we use Theorem 31 to show that our closure algorithm is also complete for sets of arbitrary XFDs, and hence that our XFD axiom system is complete for sets of arbitrary XFDs. A corollary of these results is that our chase algorithm can also be used for XFD implication for sets of arbitrary XFDs if the set is first converted to a text set.

4.2.1. Closure algorithm

We now present an algorithm for computing $(\mathbb{P}, \Sigma, P)^+$, where we recall from Definition 14 that $(\mathbb{P}, \Sigma, P)^+ = \{q \mid P \to q \in (\mathbb{P}, \Sigma)^+\}$. We denote the output of our algorithm by $clo(P)$, and note that showing soundness and completeness of our algorithm means establishing that $clo(P) = (\mathbb{P}, \Sigma, P)^+$. In this section we shall show soundness of our closure algorithm, that is $clo(P) \subseteq (\mathbb{P}, \Sigma, P)^+$. Also, without loss of generality, we assume for the rest of the article that Σ contains no redundant XFD, that is there does not exist an XFD $\sigma \in \Sigma$ such that $(\mathbb{P}, \Sigma) \equiv (\mathbb{P}, \Sigma - \{\sigma\})$.

Algorithm 1.

INPUT: A set of XFDs Σ, a downward closed set of paths $\mathbb{P}$ and a subset $P = \{p_1, \ldots, p_n\}$ of $\mathbb{P}$.
OUTPUT: $clo(P)$

$clo(P) := \{p_1, \ldots, p_n, \rho\}$;
$Unused := \Sigma$;
for each $q \in clo(P) - \{\rho\}$ **do**
  if $Last(q) \in \mathbf{E}$ **then**
    $clo(P) := clo(P) \cup \{q' \mid q' \succ q\}$;
  end if
end for
for each $q \in clo(P)$ **do**
  $clo(P) := clo(P) \cup \{q' | q = Parent(q')$ and $Last(q') \in \mathbf{A}\}$;
end for
repeat
  for each XFD $r_1, \ldots, r_k \to r_{k+1}$ in $Unused$ **do**
    if (a) there exist paths $q_1, \ldots, q_k \in clo(P)$ such that for all $i \in [1, k]$, $r_i \cap r_{k+1} \succeq q_i$ and $(q_i \succeq r_i$ or $q_i \succeq r_{k+1})$
    or (b) for all $i \in [1, k]$, $r_i \cap r_{k+1} = \rho$ or $r_i \in clo(P)$ **then**
      $clo(P) := clo(P) \cup \{r_{k+1}\}$;
      if $Last(r_{k+1}) \in \mathbf{E}$ **then**
        for each $q \succeq r_{k+1}$ **do**
          $clo(P) := clo(P) \cup \{q' \mid q = Parent(q')$ and $Last(q') \in \mathbf{A}\}$;
        end for
        $clo(P) := clo(P) \cup \{q \mid q \succ r_{k+1}\}$;
      end if
      $Unused := Unused - \{r_1, \ldots, r_k \to r_{k+1}\}$;
    end if
  end for
until no more changes can be made to $clo(P)$

We now show that our closure algorithm is sound.

Lemma 34. *If $\mathbb{P}$ is a set of paths, $P \subseteq \mathbb{P}$, Σ a set of XFDs such that $\Sigma \models \mathbb{P}$, then $clo(P) \subseteq (\mathbb{P}, \Sigma, P)^+$.*

Proof. We show the result by induction on the number of iterations in computing $clo(P)$ by Algorithm 1. At Line 1, $clo(P)$ contains $p_1, \ldots, p_n$ and ρ, and each of the paths $p_1, \ldots, p_n$ is in $(\mathbb{P}, \Sigma, P)^+$ by Axiom A1, and $\rho \in (\mathbb{P}, \Sigma, P)^+$ by Axiom A8. At Line 5, the result follows by Axiom A6 and at Line 9 by Axioms A3 and A7. So at the commencement of the repeat loop, $clo(P) \subseteq (\mathbb{P}, \Sigma, P)^+$.

Let $clo(P)_j$ denote the computation of $clo(P)$ after iteration j of the repeat loop. Assume then that the inductive hypothesis is true after iteration $j-1$, that is $clo(P)_{j-1} \subseteq (\mathbb{P}, \Sigma, P)^+$. If r_{k+1} is added to $clo(P)_{j-1}$ because of Line 13 (a), then $r_{k+1} \in (\mathbb{P}, \Sigma, P)^+$ because of Axiom A5.

Suppose instead that r_{k+1} is added to $clo(P)_{j-1}$ as a result of Line 13 (b). If for every r_i, $r_i \in clo(P)_{j-1}$, then since Axioms A1–A3 are the same as Armstrongs's axioms for FDs and $r_i \in (\mathbb{P}, \Sigma, P)^+$ by the inductive hypothesis, we deduce that $r_{k+1} \in (\mathbb{P}, \Sigma, P)^+$ and from Axiom A3 and so the inductive hypothesis again holds. If instead for every r_i, $r_i \cap r_{k+1} = \rho$, then the result holds by Axiom A4. Lastly, suppose that there exists an r_i such that $r_i \in clo(P)_{j-1}$ and $r_i \cap r_{k+1} \neq \rho$. In this case, by reindexing the paths if necessary, we can write $r_1, \ldots, r_k \to r_{k+1}$ as $r_1, \ldots, r_t, r_{t+1}, \ldots, r_k \to r_{k+1}$, $t < i$, such that

$$r_1 \cap r_{k+1} = \rho, \ldots, r_t \cap r_{k+1} = \rho, \tag{9}$$

and

$$r_{t+1} \cap r_{k+1} \neq \rho, \ldots, r_k \cap r_{k+1} \neq \rho, \tag{10}$$

and

$$r_{t+1} \in clo(P)_{j-1}, \ldots, r_k \in clo(P)_{j-1}. \tag{11}$$

If $r_{k+1} = \rho$ then since ρ is already in $clo(P)_{j-1}$ by Line 1, inserting it again will have no effect so assume that $r_{k+1} \neq \rho$. Since $(\mathbb{P}, \Sigma) \vdash r_1, \ldots, r_t, r_{t+1}, \ldots, r_k \to r_{k+1}$, it follows that if we let $p_1 = r_1 \cap r_{k+1}, \ldots, p_t = r_t \cap r_{k+1}$, $p_{t+1} = r_{t+1}, \ldots, p_k = r_k$ and $p'_1 = r_1, \ldots, p'_k = r_k$ and $q = r_{k+1}$ in Axiom A5, then Lemma 19 (vii) shows that $p'_i \cap q \succeq p_i$ and $p_i \succeq p'_i$ and so Axiom A5 applies and hence $(\mathbb{P}, \Sigma) \vdash r_1 \cap r_{k+1}, \ldots, r_t \cap r_{k+1}, r_{t+1}, \ldots, r_k \to r_{k+1}$. Combining this with (9) we deduce that

$$(\mathbb{P}, \Sigma) \vdash \rho, r_{t+1}, \ldots, r_k \to r_{k+1}. \tag{12}$$

Then, by Axiom A8, $(\mathbb{P}, \Sigma) \vdash r_{t+1}, \ldots, r_k \to \rho$ and then using Axiom A2 we deduce that $r_{t+1}, \ldots, r_k \to \rho, r_{t+1}, \ldots, r_k$. Combining this with (12) and applying Axiom A3 we obtain that

$$(\mathbb{P}, \Sigma) \vdash r_{t+1}, \ldots, r_k \to r_{k+1}. \tag{13}$$

Then from (11), the induction hypothesis and Axiom A3, we deduce that $(\mathbb{P}, \Sigma) \vdash P \to r_{t+1}, \ldots, r_k$, and combining this with (13) and Axiom A3, we deduce that $(\mathbb{P}, \Sigma) \vdash P \to r_{k+1}$ and so $r_{k+1} \in (\mathbb{P}, \Sigma, P)^+$ as required.

If Line 17 is executed, then since $r_{k+1} \in (\mathbb{P}, \Sigma, P)^+$, applying Axiom A6 shows that $q \in (\mathbb{P}, \Sigma, P)^+$. Then Axiom A7 shows that $(\mathbb{P}, \Sigma) \vdash q \to q'$, and combining with $q \in (\mathbb{P}, \Sigma, P)^+$ and using Axiom A3 shows that $q' \in (\mathbb{P}, \Sigma, P)^+$ as required.

Finally, if Line 19 is executed then Axiom A6 shows that $(\mathbb{P}, \Sigma) \vdash r_{k+1} \to q$, and since $r_{k+1} \in (\mathbb{P}, \Sigma, P)^+$ applying Axiom A3 shows that $q \in (\mathbb{P}, \Sigma, P)^+$ as required. This completes the proof. □

We now measure the running time of Algorithm 1. To do this, we define:

- $|p|$ is the number of labels in a path p;
- if σ is the XFD $p_1, \ldots, p_n \to p_{n+1}$, then $|\sigma| = |p_1| + \cdots + |p_{n+1}|$;
- $|P| = \sum_{p \in P} |p|$;
- $|\mathbb{P}| = \sum_{p \in \mathbb{P}} |p|$;
- $|\Sigma| = \sum_{\sigma \in \Sigma} |\sigma|$;
- $\|\sigma\|$ is the number of paths in σ;
- $\|P\|$ is the number of paths in P;
- $\|\mathbb{P}\|$ is the number of paths in $\mathbb{P}$;
- $\|\Sigma\|$ is the number of paths in Σ;
- $n(\Sigma)$ is the number of XFDs in Σ.

We then have the following result.

Theorem 35. *The running time of Algorithm* 1 *is $O(N^2)$, where $N = (\|\mathbb{P}\| + |\mathbb{P}|)(n(\Sigma)(\|\Sigma\| + |\Sigma|) + \|P\| + |P|)$.*

Proof. If σ denotes the XFD $r_1, \ldots, r_k \to r_{k+1}$ and if we let s_1 denote the total number of steps executed at Line 1, then

$$s_1 = \|P\| + 1. \tag{14}$$

Consider next the total number of steps executed in the **for** loop at Line 3, which we denote by s_3. Then in each iteration of the loop, the test at Line 4 takes $|q|$ steps and at most $|q| - 1$ paths can be added to $clo(P)$ since q contains $|q|$ labels. So, noting from (14) that the **for** loop at Line 3 can be iterated at most $s_1 - 1$ times, we deduce that

$$\begin{aligned} s_3 &= \sum_{i=1}^{\|P\|} |p_i|\big(|p_i| - 1\big) \\ &< \sum_{i=1}^{\|P\|} |p_i|^2 \\ &\leqslant \left(\sum_{i=1}^{\|P\|} |p_i|\right)^2 \\ &= |P|^2. \end{aligned} \tag{15}$$

Consider next the number of steps in the **for** loop at Line 8, which we denote by s_8. We first note from (14) and (15) that at the commencement of Line 8, $|clo(P)| = s_1 + s_3 \leqslant \|P\| + 1 + |P|^2$. If we combine this with the fact that the number of paths added at each iteration of Line 9 is bounded by $\|\mathbb{P}\|$, then we derive:

$$s_8 \leqslant \big(\|P\| + 1 + |P|^2\big)\|\mathbb{P}\| \leqslant \big(\|P\| + |P|\big)^2\|\mathbb{P}\|. \tag{16}$$

So then the total number of steps executed before the repeat loop is $s_1 + s_3 + s_8$ and we derive

$$\begin{aligned} s_1 + s_3 + s_8 &\leqslant \|P\| + 1 + |P|^2 + \big(\|P\| + |P|\big)^2\|\mathbb{P}\| \\ &\leqslant \big(\|P\| + |P|\big)^2 + \big(\|P\| + |P|\big)^2\|\mathbb{P}\| \text{ since } |P| \geqslant 1 \text{ and } \|P\| \geqslant 1 \\ &= \big(\|P\| + |P|\big)^2\big(\|\mathbb{P}\| + 1\big). \end{aligned} \tag{17}$$

We now compute the number of steps in each of the statements inside the loop. The test at Line 13(a) can be implemented by the following algorithm ($\|c\|$ is the number of paths in $clo(P)$ when the test is performed, which is bounded by $\|\mathbb{P}\|$).

```
For i = 1 to k Do
 { Match := false;
   For j = 1 to ‖c‖ Do
   {If r_i ∩ r(k+1) ⪰ q_j and (q_j ⪰ r_i or q_j ⪰ r(k+1))
      Then Match := true}
   If not Match then exit
 }
```

Testing whether $r_i \cap r_{k+1} \succeq q_j$ can be done in at most $|r_i \cap r_{k+1}| \leqslant |r_i|$ steps, $q_j \succeq r_i$ by $|q_j|$ steps and $q_j \succeq r_{k+1}$ by $|q_j|$ steps. So the number of steps for Line 13 (a), s_{13a}, is:

$$\begin{aligned} s_{13a} &= \sum_{i=1}^{k}\sum_{j=1}^{\|c\|}\big(|r_i| + 2|q_j|\big) \\ &= \sum_{i=1}^{k}\left(\sum_{j=1}^{\|c\|}|r_i| + \sum_{j=1}^{\|c\|}2|q_j|\right) \\ &\leqslant \sum_{i=1}^{k}\left(\sum_{j=1}^{\|\mathbb{P}\|}|r_i| + \sum_{j=1}^{\|\mathbb{P}\|}2|q_j|\right) \text{ since } \|c\| \leqslant \|\mathbb{P}\| \\ &= \sum_{i=1}^{k}\big(\|\mathbb{P}\||r_i| + 2|\mathbb{P}|\big) \leqslant \|\mathbb{P}\||\sigma| + 2k|\mathbb{P}| \\ &\leqslant \|\mathbb{P}\||\Sigma| + 2\|\sigma\||\mathbb{P}| \leqslant \|\mathbb{P}\||\Sigma| + 2\|\Sigma\||\mathbb{P}| \\ &\leqslant 2\big(\|\mathbb{P}\| + |\mathbb{P}|\big)\big(\|\Sigma\| + |\Sigma|\big). \end{aligned}$$

Now for Line 13(b), the number of steps is $s_{13b} = k$ and so is bounded by $\|\sigma\|$ and hence by $\|\Sigma\|$. For Line 14, the number of steps is $s_{14} = 1$ and for Line 15 the number of steps is $s_{15} = |r_{k+1}|$, which is bounded by $|\sigma|$ and hence by $|\Sigma|$. For Line 16, the number of steps is $s_{16} = |r_{k+1}|$, which is bounded by $|\sigma|$ and hence by $|\Sigma|$. For Line 17, the number of steps is s_{17} which is bounded by $\|\mathbb{P}\|$ and for Line 19, $s_{19} = |r_{k+1}|$, which is bounded by $|\sigma|$ and thus by $|\Sigma|$. So the total number of steps for a single iteration of the repeat loop is $s_{13a} + s_{13b} + s_{14} + s_{15} + s_{16} \times s_{17} + s_{19}$, which is $O((\|\mathbb{P}\| + |\mathbb{P}|)(\|\Sigma\| + |\Sigma|))$. Since the loop can execute at most $n(\Sigma)$ times, the total steps for the loop is $O(n(\Sigma)(\|\mathbb{P}\| + |\mathbb{P}|)(\|\Sigma\| + |\Sigma|))$, and combining this with (17) the total number of steps for the algorithm is $O((\|\mathbb{P}\| + |\mathbb{P}|)(n(\Sigma)(\|\Sigma\| + |\Sigma|) + (\|P\| + |P|)^2))$, which is $O(N^2)$ where $N = (\|\mathbb{P}\| + |\mathbb{P}|)(n(\Sigma)(\|\Sigma\| + |\Sigma|) + \|P\| + |P|)$. □

4.2.2. Completeness of the closure algorithm and axiom system

In the previous section, we showed that our closure algorithm (Algorithm 1) is sound for sets of arbitrary XFDs. In this section, we shall show that it is also complete for sets of text XFDs. To do this, we define a chase procedure for XML trees that is an extension of the classical chase algorithm for relations [1]. We note that the chase algorithm presented here is quite different from the one presented by Kot and White [13], since our method uses an XML tree rather than a relational tableau. Also, without any loss of generality we assume that every XFD in Σ contains only a single path on the r.h.s. of the XFD.

Algorithm 2.

INPUT: $\mathbb{P}, \Sigma, \mathbb{T}$; where $\Sigma \models \mathbb{P}$ and $\mathbb{T} \models \mathbb{P}$.

OUTPUT: A complete tree $\bar{\mathbb{T}}$ such that $\bar{\mathbb{T}} \models \Sigma$.

```
1:  T̄ := T;
2:  repeat
3:    for each p_1, ..., p_n → p_{n+1} in Σ do
4:      if there exist nodes v_1 ∈ [[p_1]], ..., v_{n+1} ∈ [[p_{n+1}]],
        v̄_1 ∈ [[p_1]], ..., v̄_{n+1} ∈ [[p_{n+1}]] in T̄ such that v_1, ..., v_{n+1} and v̄_1, ..., v̄_{n+1} are cn-sets;
        and for all i ∈ [1, n], val(v_n) = val(v̄_n);
        and val(v_{n+1}) ≠ val(v̄_{n+1}) then
5:        % p_1, ..., p_n → p_{n+1} is violated %
6:        if Last(p_{n+1}) ∉ E then
7:          if val(v_{n+1}) > val(v̄_{n+1}) then
8:            val(v_{n+1}) := val(v̄_{n+1})
9:          else
10:           val(v̄_{n+1}) := val(v_{n+1})
11:         end if
12:       else
13:         v := v_{n+1};
14:         v̄ := v̄_{n+1};
15:         repeat
16:           attach the subtrees rooted at each child node of v̄ to v;
17:           if there are two child attribute nodes of v with the same label then
18:             if the two nodes have the same val then
19:               delete the larger node % recall that the set of nodes V is ordered%
20:             else
21:               delete the node whose val is larger;
22:             end if
23:             v* := v̄;
24:             v := parent(v);
25:             v̄ := parent(v̄);
26:           end if
27:         until v = v̄
28:         delete v*;
29:       end if
30:     end if
31:   end for
32: until no more changes can be made to T̄
```

The algorithm works as follows. Suppose that an XFD $p_1, \ldots, p_n \to p_{n+1}$ is violated in $\bar{\mathbb{T}}$ and v and v' are two nodes in $[\![p_{n+1}]\!]$ that cause the violation. If p_{n+1} does not end with an element label, then we modify the *val* of the node in the pair $v, \bar{v}$ with the maximum *val* to the minimum *val*, and so $p_1, \ldots, p_n \to p_{n+1}$ is then satisfied. This is similar to the relational chase. If instead p_{n+1} ends in an element label, we reattach the descendants of $\bar{v}$ to v, delete $\bar{v}$ and repeat the process for

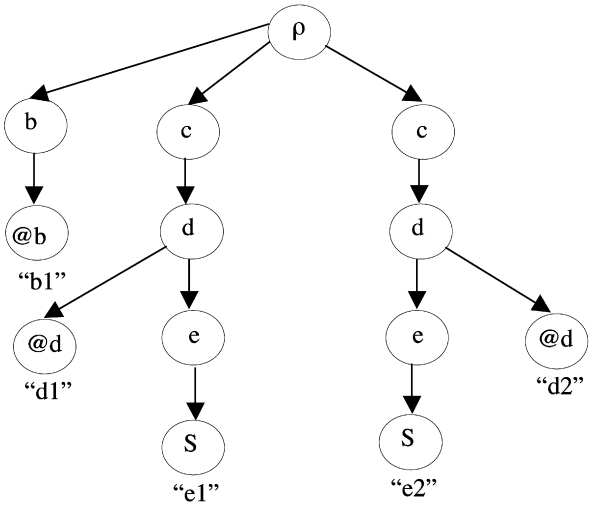

Fig. 9. Initial XML tree used in the chase.

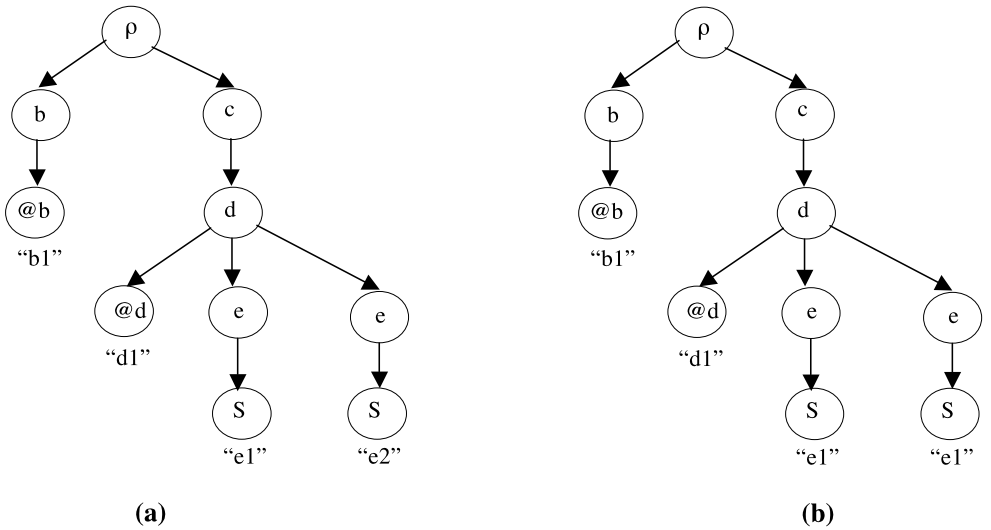

Fig. 10. XML trees during the chase.

each of the ancestors of v and $\bar{v}$. Once again, this procedure results in $p_1, \ldots, p_n \to p_{n+1}$ being satisfied. This second step has no analogue in the relational chase, but is similar to the merge rule in the chase technique used by Buneman et al. in their work on the implication of XML keys [11].

To illustrate Algorithm 2, suppose that $\Sigma = \{\rho.\mathtt{b}.\mathtt{@b} \to \rho.\mathtt{c}.\mathtt{d},\ \rho.\mathtt{c}.\mathtt{d}.\mathtt{@d} \to \rho.\mathtt{c}.\mathtt{d}.\mathtt{e}.\mathtt{S}\}$ and the initial XML tree is shown in Fig. 9. The result of applying the XFD $\rho.\mathtt{b}.\mathtt{@b} \to \rho.\mathtt{c}.\mathtt{d}$ is shown in (a) of Fig. 10, and the result of then applying the XFD $\rho.\mathtt{c}.\mathtt{d}.\mathtt{@d} \to \rho.\mathtt{c}.\mathtt{d}.\mathtt{e}.\mathtt{S}$ is shown in (b).

We note two important features of Algorithm 2. The first is that nodes are never added during the chase, the only allowable operations in the chase are changing the *val* of a node, deleting a node and reattaching a node. The second feature is that a text node can only have its *val* changed or be reattached, an attribute node can either be deleted or have its *val* changed, and an element node can only be deleted or reattached.

We now establish some fundamental properties of Algorithm 2. Also, to emphasize the dependence of the result of the chase on the input parameters, for the rest of the article we will denote the final tree generated in the chase by $\mathit{Chase}(\mathbb{T}, \mathbb{P}, \Sigma)$, rather than $\bar{\mathbb{T}}$.

Lemma 36. *Let $\mathbb{P}$ be a set of paths, $\mathbb{T}$ an XML tree such that $\mathbb{T} \models \mathbb{P}$ and Σ a set of XFDs such that $\Sigma \models \mathbb{P}$. Then*:

(i) *Algorithm* 2 *always terminates.*
(ii) *$\mathit{Chase}(\mathbb{T}, \mathbb{P}, \Sigma)$ is complete.*

(iii) *Chase*$(\mathbb{T}, \mathbb{P}, \Sigma) \models \Sigma$.
(iv) *Algorithm* 2 *possesses the Church–Rosser property.*
(v) *If* $\Sigma \equiv \Sigma'$, *then Chase*$(\mathbb{T}, \mathbb{P}, \Sigma) = Chase(\mathbb{T}, \mathbb{P}, \Sigma')$.

Proof. See Appendix A. □

Property (iv) in the above lemma, the Church–Rosser property [23], means that the XML tree generated by Algorithm 2 is independent of the sequence in which the XFDs are applied, and so the resulting XML tree *Chase*$(\mathbb{T}, \mathbb{P}, \Sigma)$ is unique.

We now define a special initial tree for testing XFD satisfaction. In the relational setting, using the chase algorithm for testing the satisfaction of an FD $X \to A$ requires a special initial tableau [1]. This tableau has two rows, and the variables are equal in the columns of X attributes but different for all other attributes in the schema. We use a similar idea for testing the satisfaction of an XFD $P \to q$. The initial tree has two nodes for every path $p \in \mathbb{P}$ except ρ, and the *val*'s for the two nodes in $[\![p]\!]$ have the same *val* if $p \in P$, and different *val*'s otherwise. The following algorithm constructs such a tree top down.

Algorithm 3.
INPUT: A set of paths $\mathbb{P} = \{p_1, \ldots, p_N\}$, a set of text paths $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{P}$.
OUTPUT: An initial XML tree $\mathbb{T}_P$.

order $\mathbb{P}$ so that if $p_i \succ p_j$ then $i < j$% hence $p_1 = \rho$%;
$\mathbb{T}_P := v_\rho$;
for $i = 2$ to N **do**
  if $Parent(p_i) = \rho$ **then**
    if $Last(p_i) \in$ **A then**
      create one child node of v_ρ with label $Last(p_i)$
    else
      create two child nodes of v_ρ with label $Last(p_i)$
    end if
  else
    for each node v_i in $[\![Parent(p_i)]\!]$ **do**
      create one child node of v_i with label $Last(p_i)$;
    end for
  end if
  if $Last(p_i) \notin$ **E then**
    if $p_i \in P$ **then**
      assign the nodes in $[\![p_i]\!]$ the same *val*
    else
      assign the nodes in $[\![p_i]\!]$ different *vals*
    end if
  end if
end for

For example, if $\mathbb{P} = \{\rho,\ \rho.\mathtt{a},\ \rho.\mathtt{a.b},\ \rho.\mathtt{a.b.c},\ \rho.\mathtt{a.@\#},\ \rho.\mathtt{a.b.@b},\ \rho.\mathtt{a.b.S},\ \rho.\mathtt{a.b.c.S}\}$ and $P = \{\rho.\mathtt{a.@\#},\ \rho.\mathtt{a.b.S}\}$, then the initial tree to be used in the chase, $\mathbb{T}_P$, is shown in Fig. 11. We note that it follows directly from Algorithm 3 that $\mathbb{T}_P \models \mathbb{P}$ and $\mathbb{T}_P$ is complete w.r.t. $\mathbb{P}$. We now use $\mathbb{T}_P$ and the chase algorithm to derive important properties of the closure algorithm.

Firstly, we denote by $\underset{P}{\Sigma}$ the set of XFDs generated by $clo(P)$ in Algorithm 1, that is:

$$\underset{P}{\Sigma} \stackrel{\text{def}}{=} \big\{ P \to q \,\big|\, q \in clo(P) \big\}. \tag{18}$$

Our next result is fundamental to establishing the completeness of our closure algorithm, both for text paths and then for arbitrary paths. It shows that given a set of XFDs Σ, a set of text paths P and the initial tree $\mathbb{T}_P$ just defined, if we chase w.r.t. the set of paths in $clo(P)$ then the resulting tree will satisfy Σ but violate any XFD not in $clo(P)$. From this result, it follows immediately that our closure algorithm is complete for set of text paths.

Theorem 37. *If* $\mathbb{P}$ *is a set of paths and* Σ *a set of text XFDs such that* $\Sigma \models \mathbb{P}$, *and* P *is a text subset of* $\mathbb{P}$, *then*:

(i) *Chase*$(\mathbb{T}_P, \mathbb{P}, \underset{P}{\Sigma}) \models \Sigma$.
(ii) *If* $q \notin clo(P)$ *then Chase*$(\mathbb{T}_P, \mathbb{P}, \underset{P}{\Sigma}) \not\models P \to q$.

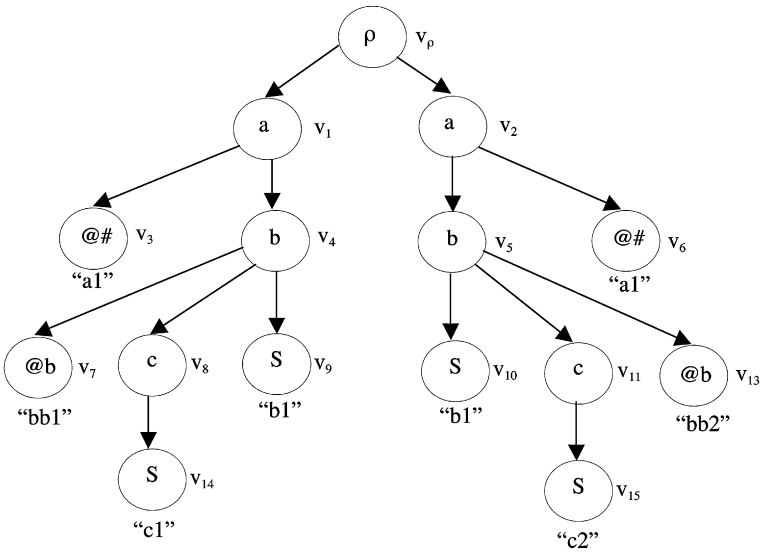

Fig. 11. Initial XML tree to be used in the chase.

Proof. See Appendix A. □

The important thing to note in this result is that we use $\underset{P}{\Sigma}$ as input to the chase algorithm, rather than Σ, and so (i) does not follow from Lemma 36. We now show that the closure algorithm is complete for sets of text XFDs.

Theorem 38. *If $\mathbb{P}$ is a set of paths and Σ a set of text XFDs such that $\Sigma \models \mathbb{P}$, and P is a text subset of $\mathbb{P}$, then $(\mathbb{P}, \Sigma, P)^+ = clo(P)$.*

Proof. Lemma 34 showed that $clo(P) \subseteq (\mathbb{P}, \Sigma, P)^+$. The fact that $(\mathbb{P}, \Sigma, P)^+ \subseteq clo(P)$ follows since if $q \notin clo(P)$, then by Theorem 37, $Chase(\mathbb{T}_P, \mathbb{P}, \underset{P}{\Sigma}) \models \Sigma$ but $Chase(\mathbb{T}_P, \mathbb{P}, \underset{P}{\Sigma}) \not\models P \to q$, and so $q \notin (\mathbb{P}, \Sigma, P)^+$. □

We now discuss briefly why the chase technique used in establishing Theorem 38 is only applicable for a set of text XFDs, and not for arbitrary XFDs. The reason is that the crucial feature of the chase technique is that in the initial tree $\mathbb{T}_P$, $[\![p]\!]$ contains two nodes for every $p \in \mathbb{P}$ and the *val* of the two nodes are the same if and only if $p \in P$. One can do this if Σ is set of text XFDs, since we have value equality for text and attribute nodes. However, we cannot do this if p ends an element label since node equality is used for such paths, and so the *val*'s of the two nodes in $[\![p]\!]$ must be different.

The first corollary below shows the soundness of the XFD axiom system for sets of text XFDs, and the second corollary shows that the chase provides an alternative method of computing XFD implication for sets of text XFDs (and also for an arbitrary set of XFDs by converting it to a set of text XFDs, as will soon be shown).

Corollary 39. *Axioms* A1–A8 *are sound and complete for sets of text XFDs.*

Proof. Soundness has already been shown in Theorem 23. For completeness, let σ be any text XFD $P \to q$, where $P \cup \{q\} \subseteq \mathbb{P}$, such that $(\mathbb{P}, \Sigma) \vdash \sigma$. So $q \in (\mathbb{P}, \Sigma, P)^+$ by definition and thus $q \in clo(P)$ from the theorem. However the proof of Lemma 34 shows that every step in the closure algorithm corresponds to an application of the XFD axioms using Σ, and so if $p \in clo(P)$ then $(\mathbb{P}, \Sigma) \overset{d}{\vdash} \sigma$, which establishes completeness. □

Corollary 40. *If $\mathbb{P}$ is a set of paths and σ is the XFD $P \to q$ and Σ and σ are a set of text XFDs and a single text XFD respectively and $\Sigma \models \mathbb{P}$ and $\sigma \models \mathbb{P}$, then the following are equivalent*:

(i) $(\mathbb{P}, \Sigma) \vdash \sigma$;
(ii) $Chase(\mathbb{T}_P, \mathbb{P}, \Sigma) \models \sigma$;
(iii) $Chase(\mathbb{T}_P, \mathbb{P}, \underset{P}{\Sigma}) \models \sigma$.

Proof. (i) ⇒ (ii) follows by (iii) of Lemma 36.

For (ii) ⇒ (iii), it suffices to show that if $\mathbb{T} = Chase(\mathbb{T}_P, \mathbb{P}, \Sigma)$ and $\bar{\mathbb{T}} = Chase(\mathbb{T}_P, \mathbb{P}, \underset{P}{\Sigma})$, then $\mathbb{T} = \bar{\mathbb{T}}$. Suppose that $\mathbb{T} \neq \bar{\mathbb{T}}$ and consider $\mathbb{T}' = Chase(\mathbb{T}, \mathbb{P}, \underset{P}{\Sigma})$. By (iii) of Lemma 36, $\mathbb{T} \models \Sigma$ and so $\mathbb{T} \models \underset{P}{\Sigma}$ since $(\mathbb{P}, \Sigma) \vdash \underset{P}{\Sigma}$ by Theorem 38 and the

definition of $\underset{P}{\Sigma}$. So when one computes $\mathit{Chase}(\mathbb{T}, \mathbb{P}, \underset{P}{\Sigma})$ using Algorithm 2 to generate $\mathbb{T}'$, the test at Line 4 fails and so no changes are made during the chase and hence $\mathbb{T}' = \mathbb{T}$. Consider then $\bar{\mathbb{T}}' = \mathit{Chase}(\bar{\mathbb{T}}, \mathbb{P}, \Sigma)$. From (i) of Theorem 37, $\bar{\mathbb{T}} \models \Sigma$ and hence $\bar{\mathbb{T}}' = \bar{\mathbb{T}}$. So if $\mathbb{T} \neq \bar{\mathbb{T}}$ then $\mathbb{T}' \neq \bar{\mathbb{T}}'$, but this contradicts (iv) of Lemma 36 and so $\mathbb{T} = \bar{\mathbb{T}}$.

For (iii) $\Rightarrow$ (i), suppose $(\mathbb{P}, \Sigma) \nvdash \sigma$. Then by Theorem 38, $q \notin \mathit{clo}(P)$ and so $\mathit{Chase}(\mathbb{T}_P, \mathbb{P}, \underset{P}{\Sigma}) \not\models \sigma$ by (ii) of Theorem 37. □

By combining these results with the results of the previous section on text XFDs, we present the main result of the article which shows that the closure algorithm, and XFD axiom system, are complete for sets of arbitrary XFDs in complete XML documents.

Theorem 41. *If $\mathbb{P}$ is a set of paths and Σ a set of XFDs such that $\Sigma \models \mathbb{P}$ and σ an XFD such that $\sigma \models \mathbb{P}$, then the following are equivalent*:

(i) $(\mathbb{P}, \Sigma) \vdash \sigma$;
(ii) $(\alpha(\mathbb{P}), \alpha(\Sigma)) \vdash \alpha(\sigma)$;
(iii) $(\alpha(\mathbb{P}), \alpha(\Sigma)) \vdash^{d} \alpha(\sigma)$;
(iv) $(\mathbb{P}, \Sigma) \vdash^{d} \sigma$.

Proof. We shall show that (i) $\Rightarrow$ (ii) $\Rightarrow$ (iii) $\Rightarrow$ (iv) $\Rightarrow$ (i). (i) $\Rightarrow$ (ii) follows from Theorem 31, (ii) $\Rightarrow$ (iii) follows from Corollary 39, (iii) $\Rightarrow$ (iv) follows from Theorem 33, (iv) $\Rightarrow$ (i) follows from the soundness of our axiom system established by Theorem 23. □

As corollaries, we show that XFD implication can be computed by the closure algorithm, or by the chase algorithm provided that the set of XFDs is first converted to a set of text XFDs.

Corollary 42. *If $\mathbb{P}$ is a set of paths and Σ a set of XFDs such that $\Sigma \models \mathbb{P}$ and σ an XFD such that $\sigma \models \mathbb{P}$ and $\sigma = P \to q$, then the following are equivalent*:

(i) $(\mathbb{P}, \Sigma) \vdash \sigma$;
(ii) $\mathit{Chase}(\alpha(\mathbb{T}_P), \alpha(\mathbb{P}), \alpha(\Sigma)) \models \alpha(\sigma)$;
(iii) $q \in \mathit{clo}(P)$.

Proof. See Appendix A. □

Regarding the time-complexity of this result, we have already shown the closure algorithm has quadratic time complexity and the following example shows that time-complexity of the chase in (ii) can be exponential in the size of the input.

Example 43. Let $\mathbb{P} = \{p_1, \ldots, p_n\}$ be an arbitrary set of n paths, let $P = \{p_1, \ldots, p_k\}, k < n$, let $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ be a set of XFDs consistent with $\mathbb{P}$ such that every XFD in Σ contains $k+1$ paths. Then it follows that the tree $\mathbb{T}_{\alpha(P)}$ generated from Algorithm 3 contains 2 nodes for every path on the l.h.s. of an XFD in Σ. Hence for every XFD in Σ, the number of combinations of nodes to be tested to see if there is a violation is 2^k, and testing a specific combination of nodes to see if it is a cn-set takes k^2 operations. So the number of operations for every XFD is at least $k^2 2^k$, and since there are k XFDs then at least $k^3 2^k$ total steps are required for Algorithm 2.

5. DTDs and XFD implication

In this section we investigate the implication problem for XFDs in the presence of a DTD. Since in this article our focus is on data centric applications of XML where the original source data is regularly structured and complete relational data, we only investigate a restricted type of DTD that is appropriate for the applications just mentioned. In previous work we presented a very general technique for transforming a flat relation to an XML tree via a nested relation generated from an arbitrary sequence of nest operators on the original flat relation [12], and thus the class of DTDs we will define corresponds to XML data obtained by such transformations.

The class of DTDs we consider, what we call *structured* DTDs, are defined as follows.

Definition 44. A *structured* DTD is defined by $\mathbb{D} \overset{\text{def}}{=} (\mathbf{E}, F, \rho)$ where:

- $\mathbf{E}$ is a finite set of element labels.
- $\mathtt{S}$ is a symbol denoting text.
- F is a mapping from $\mathbf{E}$ to element descriptions. If $e \in \mathbf{E}$ then $F(e) = \mathtt{S}$ or $F(e) = \omega_1, \ldots, \omega_n$, where $n \geqslant 1$ and ($\omega_i = e_i$ or $\omega_i = e_i^+$ and $e_i \in \mathbf{E}$).

- Every $e \in \mathbf{E}$ can appear in at most one element description, and only once in the element description.
- $\rho \in \mathbf{E}$ and is called the *element type of the root*.

Essentially, a structured DTD is disjunction free, does not allow multiple occurrences of a label nor nested element descriptions and does not contain attributes.

For example, the DTD defined by:

$\mathbf{E}$ = {ρ,a,b,d,e}
F(ρ) = a,b$^+$
F(a) = d$^+$,e
F(b) = S

is a structured DTD, but not the DTD defined by

$\mathbf{E}$ = {ρ,a,b,d,e}
F(ρ) = (a,b)$^+$
F(a) = d$^+$,e
F(b) = S

since F(ρ)=(a,b)$^+$ is a nested element description, nor is the DTD

$\mathbf{E}$ = {ρ,a,b,d,e}
F(ρ) = a, b$^+$,a,e
F(a) = d$^+$,e
F(b) = S

structured since the element label a appears twice in the first production, and the element label e appears in two different productions.

If we make a small modification to the transformation procedure given in [12], then it is easy to verify that an XML tree generated by the modified procedure always conforms to a structured DTD. Our modification is as follows. In our original procedure, a data value d for an attribute A in a relation will be mapped to an attribute node with label A and $val = d$, but instead we map it to an element node with label A having a child text node with $val = d$. An XML tree generated in this fashion will always conform to a structured DTD because attribute names are unique in a relation and so in the DTD generated by the transformation procedure of [12] an element label will appear at most once, the disjunction of attributes is not permitted in the relational model and the XML tree contains no attribute nodes.

Our assumption of a modified transformation procedure and no attributes in a structured DTD is only done to simplify the presentation and some of the results, and we could equally have retained our original transformation procedure and modified the definition of a structured DTD to allow attributes. We also note that a structured DTD is a subtype of a simple DTD as defined by Arenas and Libkin [6], since a simple DTD includes attributes and only imposes the weaker restriction that an element name can appear only once on the r.h.s. of a production rule, rather than once in the whole DTD as in our definition of a structured DTD.

Next, by augmenting the original set of XFDs with additional XFDs derived from the structure of the DTD, we show that the implication problem for a structured DTD and a set of XFDs can be converted to the implication problem for a set of XFDs alone, which is decidable and axiomatizable from the results of earlier sections.

We now give some preliminary definitions.

Definition 45. Given a structured DTD $\mathbb{D} = (\mathbf{E}, F, \rho)$ and an XML tree $\mathbb{T} = (\mathbf{E}, V, lab, ele, val, v_\rho)$, $\mathbb{T}$ *conforms* to $\mathbb{D}$, denoted by $\mathbb{T} \models \mathbb{D}$, if for all $v \in V$, if $ele(v) = [v_1, \ldots, v_n]$ then the string $lab(v_1) \cdots lab(v_n)$ is in the regular language defined by $F(lab(v))$.

We note that a structured DTD always has at least one conforming tree that is complete w.r.t. the paths of the DTD [12].

Definition 46. Given a DTD $\mathbb{D} = (\mathbf{E}, F, \rho)$, a string $p = l_1.\cdots.l_n$, where $n \geqslant 1$, is a *path* in $\mathbb{D}$ if:

- $\{l_1, \ldots, l_n\} \subseteq \mathbf{E} \cup \{S\}$;
- $l_1 = \rho$;
- for each l_i, where $i \in [2, n]$, l_i is in the alphabet of $F(l_{i-1})$.

We also denote the set of all paths in a DTD $\mathbb{D}$ by $Paths(\mathbb{D})$. We next extend the notion of an XFD being consistent with a set of paths to an XFD being consistent with a DTD.

Definition 47. An XFD $\sigma = p_1, \ldots, p_n \rightarrow p_{n+1}, \ldots, p_{n+m}$, is *consistent* with a DTD $\mathbb{D}$, denoted by $\sigma \models \mathbb{D}$, if $\{p_1, \ldots, p_{n+m}\} \subseteq \mathit{Paths}(\mathbb{D})$. A set of XFDs Σ is consistent with a DTD $\mathbb{D}$, denoted by $\Sigma \models \mathbb{D}$, if for all $\sigma \in \Sigma$, $\sigma \models \mathbb{D}$.

We now define implication by a set of XFDs and a DTD.

Definition 48. If $\mathbb{D}$ is a DTD and Σ a set of XFDs such that $\Sigma \models \mathbb{D}$, then $(\mathbb{D}, \Sigma)$ *implies* another XFD σ, denoted $(\mathbb{D}, \Sigma) \vdash \sigma$, if for every XML tree $\mathbb{T}$ such that $\mathbb{T} \models \mathbb{D}$, $\mathbb{T} \models \Sigma$ and $\mathbb{T}$ is complete w.r.t. $\mathit{Paths}(\mathbb{D})$, then $\mathbb{T} \models \sigma$. Also, we define the set of XFDs implied by $\mathbb{D}$ and Σ by $(\mathbb{D}, \Sigma)^+ = \{\sigma \mid (\mathbb{D}, \Sigma) \vdash \sigma\}$.

We now show, as for 'tree tuple' XFDs and simple DTDs in incomplete XML trees [13], the consistency problem cannot arise for 'closest node' XFDs and structured DTDs in complete trees. We first recall the definition of consistency, as proposed by Kot and White [13], except modified to include only complete trees.

Definition 49. If $\mathbb{D}$ is a structured DTD and Σ is a set of XFDs such $\Sigma \models \mathbb{D}$, then $\mathbb{D}$ and Σ are *consistent* if there exists an XML tree $\mathbb{T}$ such that $\mathbb{T} \models \mathbb{D}$, $\mathbb{T} \models \Sigma$ and $\mathbb{T}$ is complete w.r.t. $\mathit{Paths}(\mathbb{D})$.

Lemma 50. *If $D = (\mathbf{E}, F, \rho)$ is a structured DTD and Σ a set of XFDs such that $\Sigma \models \mathbb{D}$, then $\mathbb{D}$ and Σ are consistent.*

Proof. Construct a tree $\mathbb{T}$, top down, as follows. For each production rule in the DTD, construct a subtree rooted at a node that has the label of the element label on the l.h.s. of the production rule, and a single child node for each element label on the r.h.s. of the production rule. It is immediate that $\mathbb{T} \models \mathbb{D}$ and $\mathbb{T}$ is complete w.r.t. $\mathit{Paths}(\mathbb{D})$. Also, because $\mathbb{D}$ is structured and because of the construction procedure, then for every path $p \in \mathit{Paths}(\mathbb{D})$, $[\![p]\!]$ contains exactly one node. It follows immediately then from the definition of XFD satisfaction that $\mathbb{T} \models \Sigma$. □

We now present our solution to the implication problem for a set of XFDs and a structured DTD. Our method is to convert $\mathbb{D}$ and Σ to another DTD and another set of XFDs as follows.

For every $F(e)$ in $\mathbb{D}$ of the form $F(e) = \omega_1, \ldots, e_i, \ldots, \omega_n$, we change $F(e)$ to $\omega_1, \ldots, e_i^+, \ldots, \omega_n$ and add the XFD $\mathit{Parent}(p) \rightarrow p$, where $\mathit{Last}(p) = e_i$, to Σ (there can only be one such p because $\mathbb{D}$ is structured). The new DTD and the new set of XFDs are denoted by $R(\mathbb{D})$ and $R(\Sigma)$ respectively. We note that the XFDs added by this procedure are the same as those added in Theorem 4 of the article by Kot and White [13].

For example, if $\mathbb{D}$ is the DTD

$\mathbf{E} = \{\rho, \mathtt{a}, \mathtt{b}, \mathtt{d}, \mathtt{e}\}$
$F(\rho) = \mathtt{a}, \mathtt{b}^+$
$F(\mathtt{a}) = \mathtt{d}^+, \mathtt{e}$
$F(\mathtt{b}) = S$

and $\Sigma = \emptyset$, then $R(\mathbb{D})$ is given by

$\mathbf{E} = \{\rho, \mathtt{a}, \mathtt{b}, \mathtt{d}, \mathtt{e}\}$
$F(\rho) = \mathtt{a}^+, \mathtt{b}^+$
$F(\mathtt{a}) = \mathtt{d}^+, \mathtt{e}^+$
$F(\mathtt{b}) = S$

and $R(\Sigma) = \{\rho \rightarrow \mathtt{a},\ \rho.\mathtt{a} \rightarrow \rho.\mathtt{a}.\mathtt{e}\}$.

The key idea is that whenever there is a production rule that has an element label e on the r.h.s. with multiplicity one, then we replace e by e^+ in the production rule and add a XFD from the parent path of e to the path ending in e. We note that the running time of the conversion of $\mathbb{D}$ to $R(\mathbb{D})$ and Σ to $R(\Sigma)$ is linear in the number of element labels in $\mathbb{D}$.

This leads to the main result of this section, which shows that the implication problem for a structured DTD $\mathbb{D}$ and a set of FDs Σ is equivalent to the implication problem for $R(\Sigma)$ alone.

Theorem 51. *If $\mathbb{D}$ is a structured DTD and Σ a set of XFDs such that $\Sigma \models \mathbb{D}$, then $(\mathbb{D}, \Sigma)^+ = (R(\mathbb{D}), R(\Sigma))^+ = (\mathit{Paths}(R(\mathbb{D})), R(\Sigma))^+$.*

Proof. Let $\sigma = p_1, \ldots, p_n \rightarrow p_{n+1}$ be an arbitrary XFD. To establish the result, we will show that:

(i) If $\sigma \in (\mathbb{D}, \Sigma)^+$, then $\sigma \in (R(\mathbb{D}), R(\Sigma))^+$.
(ii) If $\sigma \in (R(\mathbb{D}), R(\Sigma))^+$, then $\sigma \in (\mathit{Paths}(R(\mathbb{D})), R(\Sigma))^+$.
(iii) If $\sigma \in (\mathit{Paths}(R(\mathbb{D})), R(\Sigma))^+$, then $\sigma \in (\mathbb{D}, \Sigma)^+$.

To establish (i), we show the contrapositive that if $\sigma \notin (R(\mathbb{D}), R(\Sigma))^+$, then $\sigma \notin (\mathbb{D}, \Sigma)^+$. Let $\mathbb{T}$ be a complete XML tree such that $\mathbb{T} \models R(\mathbb{D})$ and $\mathbb{T} \models R(\Sigma)$. We claim that $\mathbb{T} \models \mathbb{D}$ and $\mathbb{T} \models \Sigma$, from which the result follows immediately. By definition of $R(\Sigma)$, $\Sigma \subseteq R(\Sigma)$ and so if $\mathbb{T} \models R(\Sigma)$ then $\mathbb{T} \models \Sigma$, hence it remains to prove that if $\mathbb{T} \models R(\mathbb{D})$ then $\mathbb{T} \models \mathbb{D}$. However, this is immediate since every hedge[10] in $\mathbb{T}$ that conforms to an element description of the form e^+ can only contain one tree because $\mathbb{T} \models R(\Sigma)$, and so also conforms to $\mathbb{D}$ and so $\mathbb{T} \models \mathbb{D}$.

Consider (ii). We shall show the contrapositive that if $\sigma \notin (\mathit{Paths}(R(\mathbb{D})), R(\Sigma))^+$ then $\sigma \notin (R(\mathbb{D}), R(\Sigma))^+$. Let $\mathbb{T}$ be a complete XML tree such that $\mathbb{T} \models \mathit{Paths}(R(\Sigma))$, $\mathbb{T} \models R(\Sigma)$ but $\mathbb{T} \not\models \sigma$. We will show that by reordering sibling nodes in $\mathbb{T}$, $\mathbb{T} \models R(\mathbb{D})$ from which the result follows. Consider a non-leaf node $v \in \mathbb{T}$. Suppose that v has label e, and v has children $v_1^1, \ldots, v_n^1$ with label e_1, and $v_1^2, \ldots, v_n^2$ with label e_2 through to children $v_1^k, \ldots, v_n^k$ with label e_k. Because $\mathbb{T}$ is complete w.r.t. $\mathit{Paths}(\mathbb{D})$ and conforms to $\mathit{Paths}(R(\mathbb{D}))$, there has to be a production in $R(\mathbb{D})$ of the form $F(e) = e^+_{\theta(1)}, \ldots, e^+_{\theta(k)}$, where θ is a permutation of $\{1, \ldots, k\}$. Hence if we change the order of the children of v to $v_1^{\theta(1)}, \ldots, v_n^{\theta(1)}, \ldots, v_1^{\theta(k)}, \ldots, v_n^{\theta(k)}$, then the new subtree rooted at v conforms to $F(e)$.

Consider finally (iii). We shall show the contrapositive that if $\sigma \notin (\mathbb{D}, \Sigma)^+$ then $\sigma \notin (\mathit{Paths}(R(\mathbb{D})), R(\Sigma))^+$. Let $\mathbb{T}$ be a complete XML tree such that $\mathbb{T} \models \mathbb{D}$ and $\mathbb{T} \models \Sigma$. We shall establish that $\mathbb{T} \models \mathit{Paths}(R(\mathbb{D}))$ and $\mathbb{T} \models R(\Sigma)$, from which the result follows immediately. As noted earlier, if $\mathbb{T} \models \mathbb{D}$, then $\mathbb{T} \models R(\mathbb{D})$ and hence $\mathbb{T} \models \mathit{Paths}(R(\mathbb{D}))$, so it remains to prove that $\mathbb{T} \models R(\Sigma)$.

Suppose σ' is an arbitrary XFD in $R(\Sigma)$. Since $\Sigma \subseteq R(\Sigma)$, we can assume that $\sigma' \notin \Sigma$ and so σ' must be of the form $p \to p * e$, where $\mathit{Last}(p) = e'$ and $F_R(e') = \_, e^+, \_$.[11] Suppose to the contrary that $\mathbb{T} \not\models p \to p * e$. Then there exist $v, v' \in [\![p]\!]$ and $v_1, v_1' \in [\![p * e']\!]$ such that $\mathit{closest}(v, v_1)$ is true, $\mathit{closest}(v', v_1')$ is true, $\mathit{val}(v) = \mathit{val}(v')$ and $\mathit{val}(v_1) \neq \mathit{val}(v_1')$. Since $v, v' \in [\![p]\!]$ and $\mathit{Last}(p) = e' \in \mathbf{E}$, if $\mathit{val}(v) = \mathit{val}(v')$ then $v = v'$ from the definition of val. Then since $\mathit{closest}(v, v_1)$ is true and $\mathit{closest}(v, v_1')$ is true, $v = \mathit{parent}(v_1)$ and $v = \mathit{parent}(v_1')$ by Lemma 20 (iv). So v_1 and v_1' are both children of v, however this contradicts the fact that $\mathbb{T} \models \mathbb{D}$ since $F(e') = \_, e, \_$ and the subtree rooted at v must conform to $F(e')$. So we conclude that $\mathbb{T} \models \sigma'$ and hence $\mathbb{T} \models R(\Sigma)$. □

6. Related work

There have been several approaches to defining an XFD, which we now discuss. We do not discuss other types of XML integrity constraints, such as keys or inclusion constraints, and refer the reader to the survey by Fan [4] for details of these other types of integrity constraints.

XFDs were first introduced by Lee et al. [24], however the approach adopted was not formal and so it is difficult to compare their approach to either the 'closest node' approach, or to the 'tree tuple' approach of Arenas and Libkin [6,7]. In particular, no formal model of an XML document was given by Lee et al. and the definition of an XFD was based on informal concepts such as entities and keys.

In addition to the work previously discussed in Section 1 on investigations into the implication problem for 'tree tuple' XFDs in the presence of a DTD [6], Arenas and Libkin also made other fundamental contributions to the study of XFDs. In particular, they defined a normal form for XML documents, called XNF, and presented an algorithm for converting an unnormalized XML document into one in the normal form XNF. In subsequent work [7], Arenas and Libkin addressed the issue of providing a formal justification for XNF. Based on the classical concepts of information theory [25], they defined the concept of a 'well designed' XML schema and then showed that XNF is both a necessary and sufficient condition for an XML schema to be 'well designed'.

The issue of normal forms for XML documents, and their justification, has also been investigated by a subset of the authors in the context of strong 'closest node' XFDs [10]. This approach to justifying XML normal forms is based on formalizing the notion of redundancy elimination, a very different approach to the information theoretic approach used by Arenas and Libkin to justifying XNF. In turn, the redundancy elimination approach is an extension of our previous work that investigated the relationship between redundancy elimination and the classical relational normal forms [26]. We defined a normal form for XML documents and showed that it is a necessary and sufficient condition for the elimination of redundancy in an XML document. The normal form proposed by us differs slightly from the XNF proposed by Arenas and Libkin, but this difference only arises because our approach allows elements and text data to be mixed in an XML document, whereas the approach of Arenas and Libkin does not. If our approach was restricted to unmixed data, then the two versions of XNF would be equivalent.[12]

We note also, as mentioned in Section 1, that there is a close connection between a 'tree tuple' XFD and a weak 'closest node' XFD, and between a weak 'closest node' XFD and an FD. We have shown that a 'tree tuple' and a weak 'closest node' XFD are equivalent in XML trees which conform to a DTD and are complete w.r.t. the set of paths of the DTD, and that a weak 'closest node' XFD corresponds to a relational FD when a complete relation is first mapped to a nested relation by an arbitrary sequence of nest operations, and then directly to an XML tree [12].

[10] A hedge is a sequence of trees.

[11] _ denotes any sequence of labels.

[12] The notion of a schema being 'well designed', and that of a schema being redundancy free, have been shown to be equivalent, both for XFDs in complete XML documents and for the classical relational dependencies [12], but not for more general classes of dependencies such as transitive dependencies [27].

As mentioned in Section 1, the implication problem for 'tree tuple' XFDs has also been investigated by Kot and White [13], both without a DTD and for various classes of DTDs. Kot and White also investigated the issue of *consistency* between a DTD and a set of XFDs, which is determining whether there exists an XML tree that both conforms to the DTD and satisfies the set of XFDs. Similar to what occurs in some other classes of XML integrity constraints, such as keys [28], and keys plus inclusion constraints [29], Kot and White showed that a set of 'tree tuple' XFDs and a DTD can be inconsistent. Kot and White showed that inconsistency cannot arise for the case of simple DTDs defined by Arenas and Libkin [6], but it can occur in more general classes of DTDs such as those where the same label can appear multiple times on the r.h.s. of a production rule in the DTD. Kot and White also showed that the consistency problem can be efficiently solved for a class of DTDs that strictly includes simple DTDs, called #-DTDs, but that the consistency problem for arbitrary DTDs is NP-hard. We note that a comparison between our axiom system for XFD implication and that of Kot and White was presented in Section 3, and we showed that our axiom system is more powerful. We also note that the implication problem for XFDs has been investigated in [30].

Another approach to defining XFDs has been presented by Hartman et al. [31], using a different approach to that used in defining 'closest node' or 'tree tuple' XFDs. This approach uses subgraphs of the schema tree for the l.h.s. and r.h.s. of an XFD, rather than paths as in 'closest node' and 'tree tuple' XFDs.

A generalization of a 'tree tuple' XFD has also been presented by Yu and Jagadish [32], in the context of removing data redundancies from an XML document. The motivation for this work was the observation that neither a 'tree tuple' XFD, nor a 'closest node' XFD, can express integrity constraints involving sets of values, rather than just individual values, and a generalization of a 'tree tuple' XFD was presented which overcomes these problems. Issues such as XFD implication or normalization were not discussed in this article, since the focus of the article was on developing techniques for discovering generalized 'tree tuple' XFDs in an XML document, and hence removing data redundancies.

Several other approaches to defining an XFD have been proposed [33–36], apart from the 'tree tuple' and 'closest node' approaches. In general, these other approaches have different semantics to 'closest node' or 'tree tuple' XFDs, and the issue of developing an axiom system for XFD implication in the context of complete XML documents was not addressed. A survey of the different approaches to defining an XFD is presented in [37].

There are several other aspects of XFDs and XML normalization that have been investigated, which we now discuss. The first concerns the development of algorithms for checking an XML document for XFD satisfaction. A polynomial time algorithm for checking strong 'closest node' XFD satisfaction has been presented by Vincent and Liu [38], and checking for XFD satisfaction has also been investigated by Hartman et al. using an alternative approach to defining an XFD that was discussed previously [30]. However, to the best of our knowledge, the issues of checking for weak 'closest node' XFD satisfaction, or checking for a 'tree tuple' XFD satisfaction, have not been addressed.

The second aspect of XFDs and normalization concerns the justification for XNF. Both the work of Arenas and Libkin [7], and Vincent et al. [10], on justifying the use of XNF have assumed a native XML storage approach, and an alternative approach has been proposed by Kolahi and Libkin [39], based on the assumption that relational storage is used. Kolahi and Libkin use the 'tree tuple' approach to defining an XFD, and the notion of a 'well designed' schema discussed earlier to evaluate both XML and relational designs. They showed that the normal form XNF also guarantees a 'well designed' relational schema, under the assumption that the XML document is mapped to a relation using either the well-known inlining technique [40], or the edge technique [41]. Kolahi and Libkin also showed, in the case that an XNF design is not possible, that information redundancy can be minimized in the relational storage if the XFDs are restricted to what they term 'relative' XFDs.

The final aspect of XFDs and XML normalization concerns dependency preservation in the design of XML documents, which has been investigated by Kolahi using 'tree tuple' XFDs [42,43]. In relational databases, a classical result shows that it is not always possible to achieve a database design that is both in BCNF and dependency preserving [1]. This implies that the goals of redundancy elimination and dependency preservation are not always simultaneously achievable in relational design, since it has been shown that BCNF is a necessary and sufficient condition for the elimination of redundancy [26,44,7]. However, Kolahi showed that, in some situations, XML designs can be obtained that are both redundancy free and dependency preserving, while no redundancy free and dependency preserving design exists for the same data in the relational setting. Kolahi also derived conditions when a redundancy free and dependency preserving XML design exists, and proposed a normal form for XML that is an extension of 3NF in relational databases [1].

7. Discussion and future work

There are several aspects of the work in this article that can be extended, which we now discuss. The first relates to the choice between whether one uses node equality or value equality in defining an XFD. In this article, we have used node equality when comparing element nodes, and value equality when comparing text or attribute nodes. However, one could also define an XFD using value equality for element nodes as well as text and attribute nodes. For element nodes, this means that two nodes are value equal if the two subtrees rooted at these nodes are isomorphic and value equal on corresponding text and attribute nodes in the subtrees. This would result in different semantics for an XFD. To illustrate this, consider the XML tree in Fig. 12 and the XFD $\rho.\texttt{Dept.@Head} \rightarrow \rho.\texttt{Project}$. Adopting the definition of an XFD used in this article, this XFD expresses the semantics that the head of a department can only be associated with one project. Thus the XML tree in Fig. 12 violates $\rho.\texttt{Dept.@Head} \rightarrow \rho.\texttt{Project}$. If instead value semantics were applied when comparing

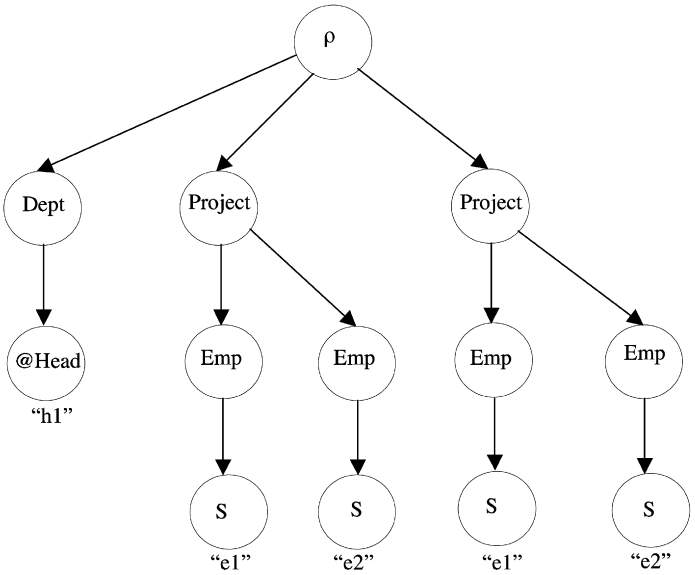

Fig. 12. An XML tree.

`Project` nodes in the definition of an XFD, then ρ . `Dept` . `@Head` $\rightarrow$ ρ . `Project` would instead express the semantics that if the head of a department was associated with two different projects, then the employees in the project must be the same. Under these semantics, the XFD ρ . `Dept` . `@Head` $\rightarrow$ ρ . `Project` would be satisfied in the XML tree in Fig. 12.

Another extension of the work in this article would be to allow more powerful path specification methods than the simple paths used in this article, such as a subset of XPath [21], or the path specification language used by Buneman et al. [11]. This would increase the expressiveness of an XFD, and also permit a much more compact notation since the notation used in this article is quite cumbersome for paths involving a large number of labels.

The 'closest node' concept used in this article also has application to understanding how to extend the other classical relational integrity constraints to XML, which is important in data centric applications of XML [14,45], where the XML data is generated from a relational database, and in XML publishing [46]. In such applications one needs to transform not only the data from relational to XML, but also the relational semantics in the form of integrity constraints. The relevance of the 'closest node' concept in this context follows from a previous result of ours [12]. This result shows that if one maps a complete relation to an XML tree by first mapping it to a nested relation using an arbitrary sequence of nest operations, and then directly to an XML tree, then two data values appear in the same tuple of the relation if and only if the corresponding nodes in the XML tree satisfy the *closest* property. This result was used to show the equivalence between an FD and a 'closest node' and, as we now demonstrate, the result can also be used to extend the other relational integrity constraints to XML.

Suppose we have a relation $R(A, B, C)$, with key (A, B). Suppose also that R is mapped to an XML document by the method previously described, and that the A values of tuples in R correspond to nodes of the path p_A, and B values correspond to nodes of a path p_B. The integrity constraint that (A, B) is a key in R requires that whenever two tuples in R have the same A and B values, then the tuples must be identical. The corresponding XML integrity constraint requires that whenever there are nodes v_A and v'_A for p_A, and nodes v_B and v'_B for p_B, such that $closest(v_A, v_B)$ and $closest(v'_A, v'_B)$ are true, and $val(v_A) = val(v'_A)$ and $val(v_B) = val(V'_B)$, then $v_A = v'_A$ and $v_B = v'_B$. The reason for this correspondance is that if the conditions are satisfied in the XML tree, then $val(v_A)$ and $val(v_B)$ must belong to the same tuple, as also do $val(v_A)$ and $val(v_B)$, and so the tuples must be the same because of the key constraint, and hence the corresponding nodes must also be the same. This is a type of uniqueness constraint, but differs in general from the keys defined by Buneman et al. [28,11], or XML Schema [47], since the *closest* property is a more general condition on the spatial connection between identifying nodes than the vertical connection used in the approach of Buneman et al. or XML Schema. It is also worth noting that this new integrity constraint is an extension, rather than simply a translation, of a relational key constraint since it applies to arbitrary XML trees, including those with duplicate and missing information. The integrity constraint can also include paths that end in an element label, rather than only paths that end in attribute or text labels as occurs for XML data generated from relational data.

Another extension of the work in this article would be to investigate the interaction of XFDs with other types of XML integrity constraints, such as XML keys as defined either in XML Schema [47], or using the approach of Buneman et al. [11]. While it has been shown that some of the classes of XML key constraints defined by Buneman et al. are special cases of a strong 'closest node' XFD [10], other classes cannot be expressed as a 'closest node' XFD, either strong or weak. For instance, given an XML tree and some path p in the tree that ends in a text label, specifying that p is key in the tree, that is no two p nodes in the document can have the same value, cannot be done using a 'closest node' XFD. The closest XFD to this key constraint is $p \rightarrow Parent(p)$. However this XFD has a different meaning, using either strong or weak 'closest node' semantics,

since it is satisfied if there are two p nodes in the tree with the same parent node. Moreover, an XML key constraint such as specifying the *vals* of a path p must be unique, interacts with XFDs since the constraint implies that the 'closest node' (either strong or weak) XFD $p \rightarrow Parent(p)$ must also be satisfied.

The issue of discovering XFDs in an XML document also warrants further investigation, though some initial work on the topic has already appeared [32,48]. The motivation for investigating this problem arises from the importance of integrity constraints in data integration, and the use of XML as the standard in data interchange [4]. Even if the source data used in integration is relational, the data is normally wrapped into XML format before integration and so, as discussed earlier, knowing the XFD integrity constraints that hold in the XML documents to be integrated can facilitate the data integration process. Unfortunately, such knowledge is often absent in practice, especially if the source data is remote, and hence the importance of recovering the original semantics from the XML data.

Another extension of our work would be to extend the investigation of the interaction between XFDs and a DTD to more general classes of DTDs than the class of structured DTDs considered in this article. For more general DTDs, the procedure given in Section 6 for structured DTDs does not capture all possible implied XFDs by the set of XFDs and DTD, as is illustrated by the following example.

Example 52. Suppose that we are given the DTD

```
<!DOCTYPE db [
  <!ELEMENT db (emp)+)>
  <!ELEMENT emp (office, phone)+, name>
  <!ELEMENT office (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
  !ELEMENT name (#PCDATA)>
  ]>
```

This DTD lists employees along with their names and their offices and phone numbers. Suppose that an XML document conforms to this DTD and satisfies the weak 'closest node' XFD `db.emp.name` $\rightarrow$ `db.emp.office`, which states that the name of an employee determines their office. Then this XFD and the DTD also imply that the XFD `db.emp.name` $\rightarrow$ `db.emp.phone` is satisfied, since conformance to the DTD requires that an `emp` element has the same number of `office` and `phone` sub-elements. This implied XFD is not captured by the results of Section 6 of this article since the DTD is not a structured DTD.

Another topic worth pursuing is to extend our approach to the more general case where there is missing information in the XML document. While this article has focused on data centric XML applications, where data is generated from a relational database, there are many other applications of XML involving more loosely structured data, such as scientific data [49], where data is normally incomplete. However, we note that the definition of a weak 'closest node' XFD applies unchanged to an XML document with missing information. In such a context, the XFD is automatically satisfied if there is a missing node for a path on the l.h.s. of the XFD. This is similar to how missing information is handled in the work by Buneman et al. on XML keys [11]. It would then be useful to develop an XFD axiom system for this more general case where data could be missing, and to then compare it with the results obtained by Kot and White [13] on axioms for 'tree tuple' XFDs, which have different semantics in this setting.

8. Conclusions

In this article we have investigated the problem of logical implication in complete XML documents for a weak 'closest node' XFD, an integrity constraint that has been shown to generalize the notion of an FD in a relational database [12]. Our main contributions have been to present an axiom system for XFD logical implication and show that the system is sound and complete, and also to derive an efficient algorithm for XFD implication.

The results of this article can be applied to automate several aspects of the design of XML documents. Firstly, our closure algorithm can be used to generate minimal sets of XFDs in the same way that the closure algorithm for FDs is used to remove redundancies from a set of relational FDs [1]. Secondly, our closure algorithm can be used to test if a set of XFDs satisfies the normal form XNF [6,7,10]. It is important to note that both of these tasks can be done efficiently because of the quadratic time performance of our closure algorithm.

Acknowledgments

We wish to thank the anonymous referees whose insightful and detailed comments helped greatly to improve the quality of this article.

Appendix A

To establish the soundness of Axiom A5 in Theorem 23, we first establish some preliminary lemmas.

Proof of Lemma 21. The proof is by induction on the number of paths in $p_1, \ldots, p_n, p_{n+1}$. Consider first the case where $n = 1$. There are three possibilities:

(a) $p_1 \succ p_2$;
(b) $p_2 \succ p_1$;
(c) $p_1 \not\succ p_2$ and $p_2 \not\succ p_1$.

If (a) holds, it follows from the fact that $\mathbb{T}$ is complete that there exists a node $v_2 \in [\![p_2]\!]$ such that $v_1 \in \mathit{aancestor}(v_2)$, and so $\mathit{closest}(v_1, v_2)$ is true from Lemma 20 (xi). Using the same reasoning, if $p_2 \succ p_1$ then $\mathit{closest}(v_1, v_2)$ is true. Suppose finally that (c) holds. Since $p_1 \cap p_2 \succeq p_1$ by Lemma 19 (vii), it follows from Lemma 20 (vii) that there is a single node x such that $x \in [\![p_1 \cap p_2]\!]$ and $x \in \mathit{aancestor}(v_1)$. Since $p_1 \cap p_2 \succeq p_2$ by Lemma 19 (viii) and $\mathbb{T}$ is complete, there exists at least one node $v_2 \in [\![p_2]\!]$ such that $x \in \mathit{aancestor}(v_2)$. It follows from the definition of *closest* that $\mathit{closest}(v_1, v_2)$ is true as required.

Next, assuming inductively that the result holds for the set of paths $p_1, \ldots, p_n$, where $n > 1$, we show that the result holds for the set $p_1, \ldots, p_n, p_{n+1}$. Consider first the set of paths $p_1 \cap p_{n+1}, \ldots, p_n \cap p_{n+1}$. These paths can be totally ordered w.r.t. $\succeq$ since each path is a prefix of p_{n+1} by Lemma 19 (vii). Then relabel the subscripts of $p_1, \ldots, p_n$ so that

$$i < j \quad \Rightarrow \quad (p_i \cap p_{n+1}) \succeq (p_j \cap p_{n+1}). \tag{a.1}$$

Let x^n_{n+1} be the node such that $x^n_{n+1} \in \mathit{aancestor}(v_n)$ and $x^n_{n+1} \in [\![p_n \cap p_{n+1}]\!]$. Such a node always exists because $p_n \cap p_{n+1} \succeq p_n$ from Lemma 19 (vii), and then applying Lemma 20 (viii). Let v_{n+1} be any node such that $v_{n+1} \in [\![p_{n+1}]\!]$ and $x^n_{n+1} \in \mathit{aancestor}(v_{n+1})$. Such a node always exists because $p_n \cap p_{n+1} \succeq p_{n+1}$ and $\mathbb{T}$ is complete. It is immediate from this construction that $\mathit{closest}(v_n, v_{n+1})$ is true, and so to complete the inductive argument we will show that

$$\text{for all } i \in [1, n-1], \quad \mathit{closest}(v_i, v_{n+1}) = \mathit{true}. \tag{a.2}$$

To do this, consider the two possible cases arising from (a.1): (a′) $(p_i \cap p_{n+1}) \succ (p_n \cap p_{n+1})$; (b′) $(p_i \cap p_{n+1}) = (p_n \cap p_{n+1})$. We consider both cases.

(a′) $(p_i \cap p_{n+1}) \succ (p_n \cap p_{n+1})$.
We first claim that

$$\text{for all } i \in [1, n-1], \quad p_i \cap p_n = p_i \cap p_{n+1}. \tag{a.3}$$

To show this, by Lemma 19 (iii) we have

$$p_i = (p_i \cap p_{n+1}) * z_1 \tag{a.4}$$

and

$$p_{n+1} = (p_i \cap p_{n+1}) * z_2 \tag{a.5}$$

and

$$z_1 \cap z_2 = \epsilon, \tag{a.6}$$

where z_1 and z_2 are label-sequences (possibly empty).

Applying Lemma 19 (iii) again, we obtain

$$p_n = (p_n \cap p_{n+1}) * z_3 \tag{a.7}$$

and

$$p_{n+1} = (p_n \cap p_{n+1}) * z_4 \tag{a.8}$$

and

$$z_3 \cap z_4 = \epsilon, \tag{a.9}$$

where z_3 and z_4 are label-sequences (possibly empty). Also, applying Lemma 19 (iii) to (a′) we obtain that

$$p_n \cap p_{n+1} = (p_i \cap p_{n+1}) * z_5, \quad \text{where } z_5 \neq \epsilon, \tag{a.10}$$

for some label-sequence z_5. So combining (a.7) and (a.10) we derive that

$$p_n = (p_i \cap p_{n+1}) * z_5 * z_3, \tag{a.11}$$

and combining (a.8) and (a.10) we deduce that

$$p_{n+1} = (p_i \cap p_{n+1}) * z_5 * z_4. \tag{a.12}$$

Thus from (a.4) and (a.11) we find that

$$p_i \cap p_n = \big((p_i \cap p_{n+1}) * z_1\big) \cap \big((p_i \cap p_{n+1}) * z_5 * z_3\big). \tag{a.13}$$

Clearly, if $z_1 = \epsilon$ then (a.13) reduces to (a.3) as claimed, so assume that $z_1 \neq \epsilon$. Comparing (a.12) and (a.5), we deduce that $z_5 * z_4 = z_2$, and combining with (a.6) we find that $z_1 \cap (z_5 * z_4) = \epsilon$. This implies, since $z_1 \neq \epsilon$ and $z_5 \neq \epsilon$ from (a.10), that z_1 and z_5 must start with different labels and so $z_1 \cap (z_5 * z_3) = \epsilon$. Thus (a.13) reduces again to (a.3) as claimed.

Next, since $closest(v_i, v_n)$ is true by the inductive hypothesis, there exists a node x_n^i such that $x_n^i \in aancestor(v_i)$, $x_n^i \in aancestor(v_n)$ and $x_n^i \in [\![p_i \cap p_n]\!]$. It follows from (a.3) that $x_n^i \in [\![p_i \cap p_{n+1}]\!]$, and so it suffices to show that $x_n^i \in aancestor(v_{n+1})$ in order to satisfy (a.2). We note that $x_n^i \in aancestor(x_{n+1}^n)$, since both nodes are *aancestors* of v_n and $p_i \cap p_{n+1} \succ p_n \cap p_{n+1}$ and applying Lemma 20 (viii). Combining this with the fact that, by definition, $x_{n+1}^n \in aancestor(v_{n+1})$ and applying Lemma 20 (ix), we derive that $x_n^i \in aancestor(v_{n+1})$. Hence $closest(v_i, v_{n+1})$ is true, and thus (a.2) also holds as claimed.

(b′) $(p_i \cap p_{n+1}) = (p_n \cap p_{n+1})$.

In this case, we first claim that $p_n \cap p_{n+1} \succeq p_i \cap p_n$. To establish this, it follows from Lemma 19 (vii) that $p_i \cap p_{n+1} \succeq p_i$, and combining this with (b′) we deduce that $p_n \cap p_{n+1} \succeq p_i$. If we then take the intersection of both sides with p_n and apply Lemma 19 (ix), we deduce that $p_n \cap p_{n+1} \succeq p_i \cap p_n$ as claimed.

Next, since $closest(v_n, v_{n+1})$ is true, there exists a node $x_{n+1}^n \in [\![p_n \cap p_{n+1}]\!]$ such that $x_{n+1}^n \in aancestor(v_n)$ and $x_{n+1}^n \in aancestor(v_{n+1})$; and since $closest(v_i, v_n)$ is true from the inductive hypothesis, there exists a node $x_n^i \in [\![p_i \cap p_n]\!]$ such that $x_n^i \in aancestor(v_i)$ and $x_n^i \in aancestor(v_n)$. This means that x_{n+1}^n and x_n^i are both *aancestors* of v_n, and since $p_n \cap p_{n+1} \succeq p_i \cap p_n$, it follows from Lemma 20 (viii) that $x_{n+1}^n \in aancestor(x_n^i)$. This means that x_{n+1}^n is also an ancestor of v_i, and hence $closest(v_i, v_{n+1})$ is true since the fact that $p_i \cap p_{n+1} = p_n \cap p_{n+1}$ from (b′) implies that $x_{n+1}^n \in [\![p_i \cap p_{n+1}]\!]$. This completes the proof.

We previously noted that the *closest* condition does not satisfy the transitivity property. The next result shows that if we place restrictions of the relationship between the relevant paths (condition (ii) of the lemma), then the transitivity rule does apply.

Lemma 53. *Let $\mathbb{P}$ be a set of paths, let $\mathbb{T}$ be an XML tree such that $\mathbb{T} \models \mathbb{P}$, let p_i, p_j, p_k be distinct paths in $\mathbb{P}$ and let $\tilde{v}_i, \tilde{v}_j, \tilde{v}_k$ be nodes in $\mathbb{T}$ such that*:

(i) $\tilde{v}_i \in [\![\tilde{p}_i]\!]$, $\tilde{v}_j \in [\![\tilde{p}_j]\!]$, $\tilde{v}_k \in [\![\tilde{p}_k]\!]$;
(ii) $\tilde{p}_j \cap \tilde{p}_k \succeq \tilde{p}_i \cap \tilde{p}_j$;
(iii) *$closest(\tilde{v}_i, \tilde{v}_j)$ is true and $closest(\tilde{v}_i, \tilde{v}_k)$ is true*;

then $closest(\tilde{v}_j, \tilde{v}_k)$ is true.

Proof. There are only three possible cases: (a′) $(\tilde{p}_i \cap \tilde{p}_k) \succ (\tilde{p}_j \cap \tilde{p}_k)$; (b′) $(\tilde{p}_i \cap \tilde{p}_k) = (\tilde{p}_j \cap \tilde{p}_k)$; (c′) $(\tilde{p}_j \cap \tilde{p}_k) \succ (\tilde{p}_i \cap \tilde{p}_k)$. We consider each case in turn.

(a′) $(\tilde{p}_i \cap \tilde{p}_k) \succ (\tilde{p}_j \cap \tilde{p}_k)$.

We claim that this case cannot occur. To verify this, it follows from Lemma 19 (vii) that $\tilde{p}_i \cap \tilde{p}_j \succeq \tilde{p}_i$, and combining with (ii) and using Lemma 19 (xi) we deduce that $\tilde{p}_j \cap \tilde{p}_k \succeq \tilde{p}_i$. However, it follows from Lemma 19 (vii) that $\tilde{p}_j \cap \tilde{p}_k \succeq p_k$ and combining with $\tilde{p}_j \cap \tilde{p}_k \succeq \tilde{p}_i$ and applying Lemma 19 (ii) we find that $\tilde{p}_j \cap \tilde{p}_k \succeq \tilde{p}_i \cap \tilde{p}_k$, which contradicts (a′) and so (a′) cannot occur.

(b′) $(\tilde{p}_i \cap \tilde{p}_k) = (\tilde{p}_j \cap \tilde{p}_k)$.

We first note that combining (b′) with (ii) we deduce that $(\tilde{p}_i \cap \tilde{p}_k) \succeq (\tilde{p}_i \cap \tilde{p}_j)$. Then since $closest(\tilde{v}_i, \tilde{v}_k)$ is true from (iii), there exists a node $x_k^i \in [\![\tilde{p}_i \cap \tilde{p}_k]\!]$ such that $x_k^i \in aancestor(\tilde{v}_i)$ and $x_k^i \in aancestor(\tilde{v}_k)$; and since $closest(\tilde{v}_i, \tilde{v}_j)$ is true from (iii), there exists a node $x_j^i \in [\![\tilde{p}_i \cap \tilde{p}_j]\!]$ such that $x_j^i \in aancestor(\tilde{v}_i)$ and $x_j^i \in aancestor(\tilde{v}_j)$. This means that x_k^i and x_j^i are both *aancestors* of $\tilde{v}_i$, and since $(\tilde{p}_i \cap \tilde{p}_k) \succeq (\tilde{p}_i \cap \tilde{p}_j)$, it follows from Lemma 20 (iv) that $x_k^i \in aancestor(x_j^i)$. This means that x_k^i is also an ancestor of $\tilde{v}_j$ from Lemma 20 (ix) since $x_j^i \in aancestor(\tilde{v}_j)$, and hence $closest(\tilde{v}_j, \tilde{v}_k)$ is true since $x_k^i \in [\![\tilde{p}_j \cap \tilde{p}_k]\!]$ from (b′).

(c′) $(\tilde{p}_j \cap \tilde{p}_k) \succ (\tilde{p}_i \cap \tilde{p}_k)$.

In this case, we first claim that $\tilde{p}_j \cap \tilde{p}_k = \tilde{p}_i \cap \tilde{p}_j$. To show this, assume to the contrary that $\tilde{p}_j \cap \tilde{p}_k \neq \tilde{p}_i \cap \tilde{p}_j$ from which it follows from (ii) that $\tilde{p}_j \cap \tilde{p}_k \succ \tilde{p}_i \cap \tilde{p}_j$, which we will show leads to a contradiction. To show this, from Lemma 19 (iii) we deduce that

$$\tilde{p}_i = (\tilde{p}_i \cap \tilde{p}_k) * z_1 \tag{a.14}$$

and

$$\tilde{p}_k = (\tilde{p}_i \cap \tilde{p}_k) * z_2 \tag{a.15}$$

where z_1 and z_2 are label-sequences such that

$$z_1 \cap z_2 = \epsilon. \tag{a.16}$$

Also, from Lemma 19 (iii) and (iv) applied to (c′) we obtain that

$$\tilde{p}_i \cap \tilde{p}_k = (\tilde{p}_j \cap \tilde{p}_k) * z'_1, \quad \text{where } z'_1 \neq \epsilon. \tag{a.17}$$

Substituting (a.17) into (a.14) and (a.19) we find that

$$\tilde{p}_i = (\tilde{p}_j \cap \tilde{p}_k) * z'_1 * z_1 \tag{a.18}$$

and

$$\tilde{p}_k = (\tilde{p}_j \cap \tilde{p}_k) * z'_1 * z_2. \tag{a.19}$$

Next, from Lemma 19 (iii) again we find that

$$\tilde{p}_i = (\tilde{p}_i \cap \tilde{p}_j) * z_3 \tag{a.20}$$

and

$$\tilde{p}_j = (\tilde{p}_i \cap \tilde{p}_j) * z_4 \tag{a.21}$$

where z_3 and z_4 are label-sequences such that

$$z_3 \cap z_4 = \epsilon. \tag{a.22}$$

Also, from Lemma 19 (iii) and (iv) applied to the assumption that $\tilde{p}_j \cap \tilde{p}_k \succ \tilde{p}_i \cap \tilde{p}_j$ we obtain that

$$\tilde{p}_i \cap \tilde{p}_j = (\tilde{p}_j \cap \tilde{p}_k) * z'_2, \quad \text{where } z'_2 \neq \epsilon. \tag{a.23}$$

Substituting (a.23) into (a.20) and (a.21) we derive that

$$\tilde{p}_i = (\tilde{p}_j \cap \tilde{p}_k) * z'_2 * z_3 \tag{a.24}$$

and

$$\tilde{p}_j = (\tilde{p}_j \cap \tilde{p}_k) * z'_2 * z_4. \tag{a.25}$$

However, comparing (a.24) and (a.18) we deduce that $z'_2 * z_4 = z'_1 * z_1$ and since z'_2 and z'_1 are not equal to the empty string, then this implies that the first label of z'_2 and the first label of z'_1 must be the same and so we can write $z'_2 = l * z''_2$ and $z'_1 = l * z''_1$, for some label-sequence z''_1. Comparing then (a.25) and (a.19) we obtain that $(\tilde{p}_j \cap \tilde{p}_k) * l$ is a common prefix of $\tilde{p}_j$ and $\tilde{p}_k$, which is a contradiction since $\tilde{p}_j \cap \tilde{p}_k$ is the longest common prefix of $\tilde{p}_j$ and $\tilde{p}_k$. So we conclude that the assumption that $\tilde{p}_j \cap \tilde{p}_k \neq \tilde{p}_i \cap \tilde{p}_j$ cannot hold, and so $\tilde{p}_j \cap \tilde{p}_k = \tilde{p}_i \cap \tilde{p}_j$ as claimed.

Next, we note that since $\tilde{p}_j \cap \tilde{p}_k \succ \tilde{p}_i \cap \tilde{p}_k$ from (c′) and $\tilde{p}_j \cap \tilde{p}_k = \tilde{p}_i \cap \tilde{p}_j$, then $\tilde{p}_i \cap \tilde{p}_j \succ \tilde{p}_i \cap \tilde{p}_k$. So since $closest(\tilde{v}_i, \tilde{v}_k)$ is true from (iii), there exists a node $x^i_k \in [\![\tilde{p}_i \cap \tilde{p}_k]\!]$ such that $x^i_k \in aancestor(\tilde{v}_i)$ and $x^i_k \in aancestor(\tilde{v}_k)$; and since $closest(\tilde{v}_i, \tilde{v}_j)$ is true from (iii), there exists a node $x^i_j \in [\![\tilde{p}_i \cap \tilde{p}_j]\!]$ such that $x^i_j \in aancestor(\tilde{v}_i)$ and $x^i_j \in aancestor(\tilde{v}_j)$. This means that x^i_k and x^i_j are both *aancestors* of $\tilde{v}_i$, and since $\tilde{p}_i \cap \tilde{p}_j \succ \tilde{p}_i \cap \tilde{p}_k$, it follows from Lemma 20 (viii) that $x^i_j \in aancestor(x^i_k)$. This means that x^i_j is also an ancestor of $\tilde{v}_k$ from Lemma 20 (ix) and because $x^i_k \in aancestor(\tilde{v}_k)$, and hence $closest(\tilde{v}_j, \tilde{v}_k)$ is true since $\tilde{p}_j \cap \tilde{p}_k = \tilde{p}_i \cap \tilde{p}_j$ implies that $x^i_j \in [\![\tilde{p}_j \cap \tilde{p}_k]\!]$. This completes the proof. □

Lemma 53 is illustrated in Fig. A.1.

The next lemma shows that the *closest* property also possesses the transitivity property in a different situation to the one identified in the previous lemma.

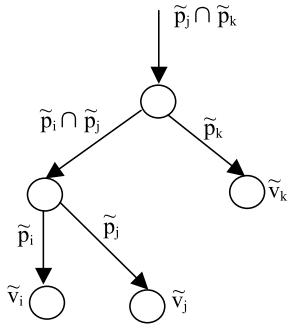

Fig. A.1. An XML tree illustrating Lemma 53.

Lemma 54. *Let $\mathbb{P}$ be a set of paths and let $\mathbb{T}$ be an XML tree such that $\mathbb{T} \models \mathbb{P}$; let $\tilde{p}_i, \tilde{p}_j, \tilde{p}_k$ be distinct paths in $\mathbb{P}$ and let $\tilde{v}_i, \tilde{v}_j, \tilde{v}_k$ be nodes in $\mathbb{T}$ such that*:

(i) $\tilde{v}_i \in [\![\tilde{p}_i]\!]$, $\tilde{v}_j \in [\![\tilde{p}_j]\!]$, $\tilde{v}_k \in [\![\tilde{p}_k]\!]$;
(ii) $\tilde{p}_i \cap \tilde{p}_j \succ \tilde{p}_i \cap \tilde{p}_k$;
(iii) *closest*$(\tilde{v}_i, \tilde{v}_j)$ *is true and closest*$(\tilde{v}_i, \tilde{v}_k)$ *is true*;

then closest$(\tilde{v}_j, \tilde{v}_k)$ *is true.*

Proof. We first claim that

$$\tilde{p}_k \cap \tilde{p}_j = \tilde{p}_i \cap \tilde{p}_j. \tag{a.26}$$

To establish this, from Lemma 19 (iii) we deduce that

$$\tilde{p}_i = (\tilde{p}_i \cap \tilde{p}_j) * z_1 \tag{a.27}$$

and

$$\tilde{p}_j = (\tilde{p}_i \cap \tilde{p}_j) * z_2 \tag{a.28}$$

where z_1 and z_2 are label-sequences such that

$$z_1 \cap z_2 = \epsilon. \tag{a.29}$$

Also, from Lemma 19 (iii), we obtain

$$\tilde{p}_k = (\tilde{p}_i \cap \tilde{p}_k) * z_3 \tag{a.30}$$

and

$$\tilde{p}_i = (\tilde{p}_i \cap \tilde{p}_k) * z_4 \tag{a.31}$$

where z_3 and z_4 are label-sequences such that

$$z_3 \cap z_4 = \epsilon. \tag{a.32}$$

Also, applying Lemma 19 (iii) and (iv) to (ii) of the lemma we obtain that

$$\tilde{p}_i \cap \tilde{p}_k = (\tilde{p}_i \cap \tilde{p}_j) * z_5, \quad \text{where } z_5 \neq \epsilon. \tag{a.33}$$

So applying (a.33) to (a.30) and to (a.31) we deduce that

$$\tilde{p}_k = (\tilde{p}_i \cap \tilde{p}_j) * z_5 * z_3 \tag{a.34}$$

and

$$\tilde{p}_i = (\tilde{p}_i \cap \tilde{p}_j) * z_5 * z_4. \tag{a.35}$$

Hence combining (a.28) and (a.34) we find that

$$\tilde{p}_k \cap \tilde{p}_j = \big((\tilde{p}_i \cap \tilde{p}_j) * z_5 * z_3\big) \cap \big((\tilde{p}_i \cap \tilde{p}_j) * z_2\big). \tag{a.36}$$

Comparing (a.27) and (a.35) we deduce that $z_1 = z_5 * z_4$, and combining this with (a.29) we deduce that $(z_5 * z_4) \cap z_2 = \epsilon$. So since $z_5 \neq \epsilon$ from (a.33), this implies that $(z_5 * z_3) \cap z_2 = \epsilon$ and so (a.36) reduces to (a.26) as claimed.

Next, since $closest(\tilde{v}_i, \tilde{v}_k)$ is true from (iii) of the lemma, there exists a node $x^i_k \in [\![\tilde{p}_i \cap \tilde{p}_k]\!]$ such that $x^i_k \in aancestor(\tilde{v}_i)$ and $x^i_k \in aancestor(\tilde{v}_k)$. Also, since $closest(\tilde{v}_i, \tilde{v}_j)$ is true from (iii) of the lemma, then there exists a node $x^i_j \in [\![\tilde{p}_i \cap \tilde{p}_j]\!]$ such that $x^i_j \in aancestor(\tilde{v}_i)$ and $x^i_j \in aancestor(\tilde{v}_j)$. So since x^i_k and x^i_j are both *aancestors* of $\tilde{v}_i$, then using (ii) of the lemma and Lemma 20 (iv) we deduce that $x^i_j \in aancestor(x^i_k)$. Combining this with the fact that $x^i_k \in aancestor(\tilde{v}_k)$ and applying Lemma 20 (viii) shows that $x^i_j \in aancestor(\tilde{v}_k)$. It follows that $closest(\tilde{v}_k, \tilde{v}_j)$ is true as claimed, since it follows from (a.26) that $x^i_j \in [\![\tilde{p}_k \cap \tilde{p}_j]\!]$. This completes the proof. □

Proof of Axiom A5. The argument is again by contra-positive, and so let $\mathbb{T}$ be a tree in Definition 22 such that $\mathbb{T} \not\models p \to q$. We first note that from Axiom A1, we can assume that for all $i \in [1, n], q \neq p_i$ and we let $p_{n+1} = q$. Also, while our definition of an XFD assumes that the paths in an XFD are distinct, it equally applies if we drop this assumption. If one does this, then it is straightforward to show that an XFD $p_1, \ldots, p_n \to q$ holds iff $RED(p_1, \ldots, p_n) \to q$ holds.

We first claim that since for all $i \in [1, n], q \neq p_i$, then for all $i \in [1, n], q \neq p'_i$. To show this, assume to the contrary that there exists a p'_i such that $p'_i = q$. So since $p'_i \cap q \succeq p_i$ by (i) of Axiom A5 and $p'_i = q$, then (i) becomes $q \succeq p_i$. Also, condition (ii) becomes $p_i \succeq q$, and combining with $q \succeq p_i$ shows that $p_i = q$. This is a contradiction, and so $q \neq p'_i$ as claimed.

Since $\mathbb{T} \not\models RED(p_1, \ldots, p_n) \to p_{n+1}$, it follows from Definition 11, and by duplicating nodes for duplicating paths in $p_1, \ldots, p_n$, that there exist nodes $v_1, \ldots, v_n, v_{n+1}$ and nodes $v'_1, \ldots, v'_n, v'_{n+1}$ in $\mathbb{T}$ such that:

(a) $v_1 \in [\![p_1]\!], \ldots, v_n \in [\![p_n]\!], v_{n+1} \in [\![p_{n+1}]\!]$ and $v'_1 \in [\![p_1]\!], \ldots, v'_n \in [\![p_n]\!], v'_{n+1} \in [\![p_{n+1}]\!]$;
(b) $v_1, \ldots, v_{n+1}$ and $v'_1, \ldots, v'_{n+1}$ are cn-sets;
(c) for all $i \in [1, n]$, $val(v_i) = val(v'_i)$;
(d) $val(v_{n+1}) \neq val(v'_{n+1})$.

Next, we modify the set of nodes $v'_1, \ldots, v'_n$ as follows. For any node v_i in the set $v_1, \ldots, v_n$, if $closest(v_i, v'_{n+1})$ is also true, then replace v'_i by v_i in the set $v'_1, \ldots, v'_n$.

We claim that this modified set of nodes is also a cn-set. To verify this, since $v_1, \ldots, v_n$ is a cn-set from (b), it suffices to show that if there are two nodes v_i and v_k in the set $v_1, \ldots, v_n$ such that $closest(v_i, v'_{n+1})$ is true, but $closest(v_k, v'_{n+1})$ is false, then $closest(v_i, v'_k)$ is true.

To prove this, we first claim that

$$p_i \cap p_{n+1} \succ p_k \cap p_{n+1}. \tag{a.37}$$

To show this, assume to the contrary that $p_k \cap p_{n+1} \succeq p_i \cap p_{n+1}$. Then since $closest(v_i, v_{n+1})$ is true from (b), and $closest(v_i, v'_{n+1})$ is true by assumption, then there exist nodes x^i_{n+1} and $\bar{x}^i_{n+1}$ in $[\![p_i \cap p_{n+1}]\!]$ which are both an *aancestor* of v_i. So from Lemma 20 (v), we conclude that $x^i_{n+1} = \bar{x}^i_{n+1}$. Next, since $closest(v_k, v_{n+1})$ is true from (b), there exists a node $x^k_{n+1} \in [\![p_k \cap p_{n+1}]\!]$ such that $x^k_{n+1} \in aancestor(v_k)$ and $x^k_{n+1} \in aancestor(v_{n+1})$. So x^k_{n+1} and x^i_{n+1} are both *aancestor's* of v_{n+1}, and combining this with the assumption that $p_k \cap p_{n+1} \succeq p_i \cap p_{n+1}$ and applying Lemma 20 (viii) shows that $x^k_{n+1} \in aancestor(x^i_{n+1})$. Thus $x^k_{n+1} \in aancestor(\bar{x}^i_{n+1})$ also because $x^i_{n+1} = \bar{x}^i_{n+1}$, and so $x^k_{n+1} \in aancestor(v'_{n+1})$ since $\bar{x}^i_{n+1} \in aancestor(v'_{n+1})$. This implies that $closest(v_k, v'_{n+1})$ is true, which is a contradiction and so we conclude that (a.37) holds as claimed.

We now apply Lemma 54 and make the following substitutions: $\tilde{p}_i \mapsto p_{n+1}, \tilde{p}_j \mapsto p_i, \tilde{p}_k \mapsto p_k; \tilde{v}_i \mapsto v'_{n+1}, \tilde{v}_j \mapsto v_i$, $v_k \mapsto v'_k$. Then (i) of Lemma 54 holds from (a), (ii) holds from (a.37), (iii), (iii) holds since $closest(v_i, v'_{n+1})$ is true by assumption and $closest(v'_k, v'_{n+1})$ follows from (b). So the conditions of Lemma 54 are satisfied and so we conclude that $closest(v_i, v'_k)$ is true and so the modified set of nodes is a cn-set as claimed.

Next, we note that since $q \neq p_i$ and $q = p_{n+1}$, then for any path p_i in $p_1, \ldots, p_n$, $p_i \neq p_{n+1}$. So from condition (ii) of Axiom A5, one of the following conditions must be satisfied:

(i) $p_i = p'_i$;
(ii) $p_i \succ p'_i$;
(iii) $p_i \succ p_{n+1}$.

We now show how to choose sets of nodes $\bar{v}_1, \ldots, \bar{v}_{n+1}$ and $\bar{v}'_1, \ldots, \bar{v}'_{n+1}$ such that:

(a′) $\bar{v}_1 \in [\![p'_1]\!], \ldots, \bar{v}_n \in [\![p'_n]\!], \bar{v}_{n+1} \in [\![p_{n+1}]\!]$ and $\bar{v}'_1 \in [\![p'_1]\!], \ldots, \bar{v}'_n \in [\![p'_n]\!], \bar{v}'_{n+1} \in [\![p_{n+1}]\!]$;
(b′) $\bar{v}_1, \ldots, \bar{v}_{n+1}$ is a cn-set;
(c′) $\bar{v}'_1, \ldots, \bar{v}'_{n+1}$ is a cn-set;
(d′) $\forall i \in [1, n]$, $val(\bar{v}_i) = val(\bar{v}'_i)$;
(e′) $val(\bar{v}_{n+1}) \neq val(\bar{v}'_{n+1})$.

It follows immediately from Definition 11 that if there exist sets of nodes with properties (a′)–(e′), then $\mathbb{T} \not\models p'_1, \ldots, p'_n \rightarrow p_{n+1}$ as required.

First, we let $\bar{v}_{n+1} = v_{n+1}$ and $\bar{v}'_{n+1} = v'_{n+1}$, and so (e′) follows from (e).

Next, we choose the nodes $\bar{v}_1, \ldots, \bar{v}_n$ and $\bar{v}'_1, \ldots, \bar{v}'_n$ as follows. Given the path p'_i, let $M(p'_i)$ denote the path p that maximizes the length of $p'_i \cap p$ w.r.t. $\succeq$, where p is any path in the set $p_1, \ldots, p_n, p_{n+1}, p'_1, \ldots, p'_{i-1}$. If there is more than one such path, then the choice of $M(p'_i)$ is arbitrary. Thus the following condition holds:

$$\forall p \in \{p_1, \ldots, p_n, p_{n+1}, p'_1, \ldots, p'_{i-1}\}, \quad p'_i \cap p \succeq p'_i \cap M(p'_i). \tag{a.38}$$

We then choose nodes in increasing index order as follows. For arbitrary $i \in [1, n]$, choose nodes $\bar{v}_i$ and $\bar{v}'_i$ as follows:

(A) If (i) holds, then $\bar{v}_i = v_i$ and $\bar{v}'_i = v'_i$.
(B) If (ii) holds, let $\bar{v}_i$ be any node in $[\![p'_i]\!]$ such that $closest(\bar{v}_i, v_m)$ is true, where v_m is the node in the cn-set $\{v_1, \ldots, v_{n+1}\}$ that is also in $[\![M(p'_i)]\!]$, and let $\bar{v}'_i = \bar{v}_i$.
(C) If (iii) holds, let $\bar{v}_i$ be any node in $[\![p'_i]\!]$ such that $closest(\bar{v}_i, v_m)$ is true, where v_m is the node in the cn-set $\{v_1, \ldots, v_{n+1}\}$ that is also in $[\![M(p'_i)]\!]$ (as in (ii)), and let $\bar{v}'_i$ be any node in $[\![p'_i]\!]$ such that $closest(\bar{v}'_i, v'_m)$ is true, where v'_m is the node in the cn-set $\{v'_1, \ldots, v'_{n+1}\}$ that is also in $[\![M(p'_i)]\!]$.

We note that since $\mathbb{T}$ is complete w.r.t. $\mathbb{P}$, then the node $\bar{v}_i$ chosen at either (B) or (C) always exists. We also note that node $\bar{v}_m$ is chosen before node $\bar{v}_i$.

Consider now the conditions (a′)–(d′) that need to be established. First, it follows from (A)–(C) that condition (a′) is satisfied. We next claim that condition (d′) holds. To verify this, if (A) holds then (d′) follows from the fact that $val(v_i) = val(\bar{v}_i)$ from (c). If (B) or (C) hold, then p_i is a strict prefix of either p'_i or p_{n+1}, and so must end with an element label. So since $val(v_i) = val(v'_i)$ from (c), then it follows from the definition of val that

$$v_i = v'_i. \tag{a.39}$$

Combining this result with (B) and (C) we deduce that $\bar{v}_i = \bar{v}'_i$, and so $\bar{v}_i = \bar{v}'_i$ and hence (d′) holds as claimed.

Since we showed earlier that (e′) holds, it remains to show that (b′) and (c′) hold. To do this, remembering that any subset of a cn-set is also a cn-set, we will show by induction that

$$v_1, \ldots, v_{n+1}, \bar{v}_1, \ldots, \bar{v}_{n+1} \text{ is a cn-set} \tag{a.40}$$

and

$$v'_1, \ldots, v'_{n+1}, \bar{v}'_1, \ldots, \bar{v}'_{n+1} \text{ is a cn-set}, \tag{a.41}$$

and hence $\bar{v}_1, \ldots, \bar{v}_{n+1}$, and $\bar{v}'_1, \ldots, \bar{v}'_{n+1}$ are also cn-sets, and thus (b′) and (c′) will hold.

Clearly, the sets $v_1, \ldots, v_{n+1}$ and $v'_1, \ldots, v'_{n+1}$ are cn-sets from (b), and so assume inductively that $v_1, \ldots, v_{n+1}, \bar{v}_1, \ldots, \bar{v}_{i-1}$ and $v'_1, \ldots, v'_{n+1}, \bar{v}'_1, \ldots, \bar{v}'_{i-1}$ are cn-sets. Then to show that $v_1, \ldots, v_{n+1}, \bar{v}_1, \ldots, \bar{v}_{i-1}, \bar{v}_i$ is a cn-set, and thus (a.40) holds, it suffices to show that

$$closest(v, \bar{v}_i) \text{ is true for any node } v \text{ in the set } \{v_1, \ldots, v_{n+1}, \bar{v}_1, \ldots, \bar{v}_{i-1}\}; \tag{a.42}$$

and to show that $v'_1, \ldots, v'_{n+1}, \bar{v}'_1, \ldots, \bar{v}'_{i-1}, \bar{v}'_i$ is a cn-set, and so (a.41) holds, it suffices to show that

$$closest(v', \bar{v}'_i) \text{ is true for any node } v' \text{ in the set } \{v'_1, \ldots, v'_{n+1}, \bar{v}'_1, \ldots, \bar{v}'_{i-1}\}. \tag{a.43}$$

We now show that both these equations hold for each of the cases (A)–(C).

(A) In this case $\bar{v}_i = v_i$ and $\bar{v}'_i = v'_i$. However, it follows immediately that $closest(v, \bar{v}_i)$ is true and $closest(v', \bar{v}'_i)$ is true since $v_1, \ldots, v_{n+1}$, $\bar{v}_1, \ldots, \bar{v}_{i-1}$ and $v'_1, \ldots, v'_{n+1}$, $\bar{v}'_1, \ldots, \bar{v}'_{i-1}$ are cn-sets by the inductive hypothesis, and so (a.42) and (a.43) both hold as required.

(B) Since node v is in the set $v_1, \ldots, v_{n+1}, \bar{v}_1, \ldots, \bar{v}_{i-1}$, then $v \in [\![p]\!]$ for some path p in the set of paths $p_1, \ldots, p_{n+1}, p'_1, \ldots, p'_{i-1}$. Consider then Lemma 53, and make the substitutions: $\tilde{p}_i \mapsto M(p'_i)$, $\tilde{p}_j \mapsto p'_i$, $\tilde{p}_k \mapsto p$; $\tilde{v}_i \mapsto v_m$, $\tilde{v}_j \mapsto \bar{v}_i$, $\tilde{v}_k \mapsto v$. With these substitutions, condition (i) of Lemma 53 becomes $v_m \in [\![M(p'_i)]\!]$ and $\bar{v}_i \in [\![p'_i]\!]$ and $v \in [\![p]\!]$. The first condition holds from (a), the second from (B) and the third from the definition of v. Condition (ii) of Lemma 53 becomes $p \cap p'_i \succeq M(p'_i) \cap p'_i$, which holds from (B) and (a.38). Condition (iii) of Lemma 53 becomes $closest(\bar{v}_i, v_m)$ is true, which holds from (B), and $closest(v, v_m)$ is true, which holds from the induction hypothesis. So the conditions of Lemma 53 are satisfied, and hence $closest(\bar{v}_i, v)$ is true and so (a.42) holds as required.

Next, suppose that (B) holds and consider (a.43). Suppose first that $v' = v'_{n+1}$, and so we need to establish that $closest(\bar{v}'_i, v'_{n+1})$ is true which, since $\bar{v}'_i = \bar{v}_i$ from (B), becomes

$$closest(\bar{v}_i, v'_{n+1}) \text{ is true.} \tag{a.44}$$

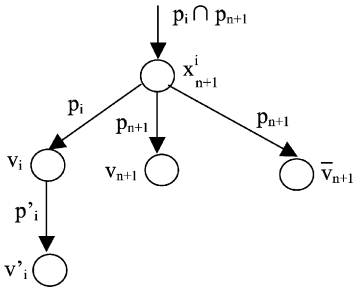

Fig. A.2. XML tree illustrating the proof of Axiom A5.

We first claim that

$$p'_i \cap p_{n+1} = p_i \cap p_{n+1}. \tag{a.45}$$

To show this, since $p_i \succ p'_i$ from (B), taking the intersection of both sides with p_{n+1} and applying Lemma 19 (ix) shows that $p_i \cap p_{n+1} \succeq p'_i \cap p_{n+1}$. For the reverse direction, since $p'_i \cap p_{n+1} \succeq p_i$ from (i) of Axiom A5, taking the intersection of both sides with p_{n+1} and applying Lemma 19 (ix) shows that $p'_i \cap p_{n+1} \succeq p_i \cap p_{n+1}$ and so (a.45) holds as claimed.

Next, since $closest(v'_i, v'_{n+1})$ is true from (b) and $v_i = v'_i$ from (a.39), then $closest(v_i, v'_{n+1})$ is true. So there exists a node $x^i_{n+1} \in [\![p_i \cap p_{n+1}]\!]$ such that $x^i_{n+1} \in aancestor(v_i)$ and $x^i_{n+1} \in aancestor(v'_{n+1})$. Also, since $closest(v_i, \bar{v}_i)$ is true since we have shown that (a.42) holds, and $p_i \succ p'_i$ from (B), then it follows from Lemma 20 (viii) that $v_i \in aancestor(\bar{v}_i)$. Combining this with the fact that $x^i_{n+1} \in aancestor(v_i)$ and applying Lemma 20 (ix) shows that $x^i_{n+1} \in aancestor(\bar{v}_i)$. It follows then that (a.44) holds as required, since it follows from (a.45) that $x^i_{n+1} \in [\![p'_i \cap p_{n+1}]\!]$. The situation is illustrated in Fig. A.2.

Suppose next that $v' \neq v'_{n+1}$. Then $v' = v'_k$ for some node v'_k in the set $v'_1, \ldots, v'_n$, or $v' = \bar{v}'_k$ some node $\bar{v}'_k$ in the set $\bar{v}'_1, \ldots, \bar{v}'_{i-1}$. So to establish (a.43), since $\bar{v}'_i = \bar{v}_i$ from (a.39), we need to show that

$$closest\big(\bar{v}_i, v'_k\big) \text{ is true for arbitrary } v'_k \text{ in } v'_1, \ldots, v'_n \tag{a.46}$$

and

$$closest\big(\bar{v}_i, \bar{v}'_k\big) \text{ is true for arbitrary } \bar{v}'_k \text{ in } \bar{v}'_1, \ldots, \bar{v}'_{i-1}. \tag{a.47}$$

Consider first (a.46). If $v'_k = v_k$, then (a.46) follows from (a.42) so assume that $v'_k \neq v_k$. We note that in this case (a.37) holds.

Next, we claim that $closest(v_i, v'_k)$ is true. To verify this, consider Lemma 54 and make the following substitutions: $\tilde{p}_i \mapsto p_{n+1}, \tilde{p}_j \mapsto p_i, \tilde{p}_k \mapsto p_k$; $\tilde{v}_i \mapsto v'_{n+1}, \tilde{v}_j \mapsto v_i, \tilde{v}_k \mapsto v'_k$. Then condition (i) of Lemma 54 becomes $v'_{n+1} \in [\![p_{n+1}]\!]$ and $v_i \in [\![p_i]\!]$ and $v'_k \in [\![p_k]\!]$, which holds from (a). Condition (ii) of Lemma 54 becomes $p_i \cap p_{n+1} \succ p_k \cap p_{n+1}$, which holds from (a.37). Finally, (iii) becomes $closest(v_i, v'_{n+1})$ is true, and $closest(v'_k, v'_{n+1})$ is true. The first condition holds since $closest(v'_i, v'_{n+1})$ is true from (b) and $v'_i = v_i$ from (a.39), and the second condition holds from (b). Thus the conditions of Lemma 54 are satisfied and so $closest(v_i, v'_k)$ is true as claimed.

We next claim that

$$p'_i \cap p_k = p_i \cap p_k. \tag{a.48}$$

We verify this by showing that

$$p'_i \cap p_k = p_i \cap p_{n+1} \tag{a.49}$$

and

$$p_i \cap p_k = p_i \cap p_{n+1}, \tag{a.50}$$

from which (a.48) follows immediately. We first establish (a.49). To do this, there are only three possible cases: $p_i \succ p_{n+1}$; $p_{n+1} \succ p_i$; $p_i \not\succ p_{n+1}$ and $p_{n+1} \not\succ p_i$.

For the first case, if $p_i \succ p_{n+1}$ then it follows from Lemma 19 (iv) that $p_i = p_i \cap p_{n+1}$ and so (a.49) becomes

$$p'_i \cap p_k = p_i. \tag{a.51}$$

Then, combining $p_i = p_i \cap p_{n+1}$ with (a.37) we conclude that

$$p_i \succ p_k \cap p_{n+1}. \tag{a.52}$$

If we then combine this result with this with the fact that $p_k \cap p_{n+1} \succeq p_k$ from Lemma 19 (vii) and apply Lemma 19 (xi), we deduce that $p_i \succeq p_k$. Combining this with the fact that $p_i \succ p'_i$ from (ii) and applying Lemma 19 (ii) we derive that

$$p_i \succeq p'_i \cap p_k. \tag{a.53}$$

Next, applying Lemma 19 (iii) we obtain that

$$p_k = (p_k \cap p_{n+1}) * z_1 \tag{a.54}$$

and

$$p_{n+1} = (p_k \cap p_{n+1}) * z_2 \tag{a.55}$$

where

$$z_1 \cap z_2 = \epsilon. \tag{a.56}$$

Also, combining (a.52) with Lemma 19 (iii) we find that

$$p_k \cap p_{n+1} = p_i * z'_1, \quad \text{where } z'_1 \neq \epsilon. \tag{a.57}$$

Substituting (a.57) into (a.54) and (a.55) we deduce that

$$p_k = p_i * z'_1 * z_1 \tag{a.58}$$

and

$$p_{n+1} = p_i * z'_1 * z_2. \tag{a.59}$$

Applying Lemma 19 (iii) again we deduce that

$$p'_i = \left(p'_i \cap p_k\right) * z_3 \tag{a.60}$$

and

$$p_k = \left(p'_i \cap p_k\right) * z_4 \tag{a.61}$$

where

$$z_3 \cap z_4 = \epsilon. \tag{a.62}$$

Also, combining (a.53) with Lemma 19 (iii) we find that

$$p'_i \cap p_k = p_i * z'_2, \tag{a.63}$$

and substituting this into (a.60) and (a.61) we derive that

$$p'_i = p_i * z'_2 * z_3 \tag{a.64}$$

and

$$p_k = p_i * z'_2 * z_4. \tag{a.65}$$

Suppose first that $z'_2 = \epsilon$. Then since $p'_i \cap p_{n+1} \succeq p_i$ from (i) of Axiom A5, combining with (a.64) and (a.59) shows that $(p_i * z_3) \cap (p_i * z'_1 * z_2) \succeq p_i$, and so $z_3 \cap (z'_1 * z_2) = \epsilon$. Since $z'_1 \neq \epsilon$ from (a.57), it follows then that either $z_3 = \epsilon$, or $z_3 \neq \epsilon$ and the first label in z_3 and z'_1 are different. However, combining (a.64) and (a.65) we deduce that $p'_i \cap p_k = (p_i * z_3) \cap (p_i * z'_1 * z_1)$ and since either $z_3 = \epsilon$, or $z_3 \neq \epsilon$ and z_3 and z'_1 start with different labels, then $z_3 \cap (z'_1 * z_1) = \epsilon$ and hence $p'_i \cap p_k = p_i$ and so (a.51) and (a.49) hold as claimed.

If instead $z'_2 \neq \epsilon$, then since $p'_i \cap p_{n+1} \succeq p_i$ from (i) of Axiom A5, combining with (a.64) and (a.59) shows that $(p_i * z'_2 * z_3) \cap (p_i * z'_1 * z_2) \succeq p_i$. So since $z'_1 \neq \epsilon$ and $z'_2 \neq \epsilon$, this implies that z'_1 and z'_2 must have different first labels. However from (a.64) and (a.65) we deduce that $p'_i \cap p_k = (p_i * z'_2 * z_3) \cap (p_i * z'_1 * z_1)$, and since z'_1 and z'_2 have different first labels then $(z'_2 * z_3) \cap (z'_1 * z_1) = \epsilon$ and so $p'_i \cap p_k = p_i$ and so (a.51) and (a.49) again hold. This completes the proof of (a.49) for the case where $p_i \succ p_{n+1}$.

Consider then the second case where $p_{n+1} \succ p_i$. Then since $closest(v_i, v_{n+1})$ is true from (b), then applying Lemma 20 (iv) shows that $v_{n+1} \in aancestor(v_i)$. Similarly, since $closest(v_i, v'_{n+1})$ is true since $v_i = v'_i$ from (a.39) and $closest(v'_i, v'_{n+1})$ holds from (b), then $v'_{n+1} \in aancestor(v_i)$. However, since both v_{n+1} and v'_{n+1} are nodes in $[\![p_{n+1}]\!]$ from (a), then it follows from Lemma 20 (v) that $v_{n+1} = v'_{n+1}$, and hence $val(v_{n+1}) = val(v'_{n+1})$. However this contradicts (d), and so we conclude that the second case cannot arise.

Consider then the final case where $p_i \not\succ p_{n+1}$ and $p_{n+1} \not\succ p_i$. In this case, from Lemma 19 (iii), we deduce that

$$p_i = (p_i \cap p_{n+1}) * z_1 \tag{a.66}$$

and

$$p_{n+1} = (p_i \cap p_{n+1}) * z_2 \tag{a.67}$$

where

$$z_1 \neq \epsilon \text{ and } z_2 \neq \epsilon \text{ and } z_1 \cap z_2 = \epsilon. \tag{a.68}$$

Also, since $p_i \succ p'_i$ from (B), we deduce from Lemma 19 (iii) and (iv) that

$$p'_i = p_i * z_3, \quad \text{where } z_3 \neq \epsilon, \tag{a.69}$$

and combining with (a.66) we deduce that

$$p'_i = (p_i \cap p_{n+1}) * z_1 * z_3. \tag{a.70}$$

From Lemma 19 (iii) again, we deduce that

$$p_k = (p_k \cap p_{n+1}) * z_4 \tag{a.71}$$

and

$$p_{n+1} = (p_k \cap p_{n+1}) * z_5, \tag{a.72}$$

where

$$z_4 \cap z_5 = \epsilon. \tag{a.73}$$

However, from (a.37) and Lemma 19 (iii) and (iv) we deduce that

$$p_k \cap p_{n+1} = (p_i \cap p_{n+1}) * z_6, \quad \text{where } z_6 \neq \epsilon. \tag{a.74}$$

Applying (a.74) to (a.71) and to (a.72) we obtain

$$p_k = (p_i \cap p_{n+1}) * z_6 * z_4 \tag{a.75}$$

and

$$p_{n+1} = (p_i \cap p_{n+1}) * z_6 * z_5. \tag{a.76}$$

Combining then (a.70) and (a.75) we find that

$$p'_i \cap p_k = \big((p_i \cap p_{n+1}) * z_1 * z_3\big) \cap \big((p_i \cap p_{n+1}) * z_6 * z_4\big). \tag{a.77}$$

However, comparing (a.76) and (a.67) we deduce that $z_6 * z_5 = z_2$ and so, from (a.68), we find that $z_1 \cap (z_6 * z_5) = \epsilon$. This implies, since $z_1 \neq \epsilon$ from (a.68) and $z_6 \neq \epsilon$ from (a.74), that $(z_1 * z_3) \cap (z_6 * z_4) = \epsilon$, and so (a.77) reduces to (a.49) as claimed.

Next, to establish (a.50), combining (a.66) and (a.71) we deduce that

$$p_i \cap p_k = \big((p_i \cap p_{n+1}) * z_1\big) \cap \big((p_i \cap p_{n+1}) * z_6 * z_4\big). \tag{a.78}$$

However, since $z_1 \cap (z_6 * z_5) = \epsilon$ as noted earlier, and $z_1 \neq \epsilon$ from (a.68) and $z_6 \neq \epsilon$ from (a.74), then $z_1 \cap (z_6 * z_4) = \epsilon$ and so (a.78) reduces to (a.50) as claimed, and thus (a.48) also holds as claimed.

Next, since $v_i = v'_i$ from (a.39) and $\mathit{closest}(v'_i, v'_k)$ is true from (b), then $\mathit{closest}(v_i, v'_k)$ is true and so there exists a node $x^i_k \in [\![p_i \cap p_k]\!]$ such that $x^i_k \in \mathit{aancestor}(v_i)$ and $x^i_k \in \mathit{aancestor}(v'_k)$. However, since $\mathit{closest}(\bar{v}_i, v_i)$ is true since we have shown that $v_1, \ldots, v_{n+1}, \bar{v}_1, \ldots, \bar{v}_{i-1}, \bar{v}_i$ is a cn-set from (a.42), and $p_i \succ p'_i$ from (B), then $v_i \in \mathit{aancestor}(\bar{v}_i)$ from Lemma 20 (iv). Combining this with the fact that $x^i_k \in \mathit{aancestor}(v_i)$ and using Lemma 20 (ix) shows that $x^i_k \in \mathit{aancestor}(\bar{v}_i)$. Thus $\mathit{closest}(\bar{v}_i, v'_k)$ is true since it follows from (a.48) that $x^i_k \in [\![p'_i \cap p_k]\!]$, and so (a.46) holds as claimed. The situation is illustrated in Fig. A.3.

Consider next (a.47). In this case, $\bar{v}'_k$ could have been chosen from either (A), (B) or (C). If it was chosen from (A), then from (A) we find that $\bar{v}'_k = v'_k$, and so (a.47) follows from (a.46). If instead $\bar{v}'_k$ is chosen from either (B) or (C), then $\bar{v}'_k = \bar{v}_k$ and so (a.47) follows from (a.42). This completes (a.43) for the case of (B).

(C) We first note that since $p_i \succ p_{n+1}$ from (C), then p_i must end in an element label and so if $\mathit{val}(v_i) = \mathit{val}(v'_i)$, then (a.39) again holds.

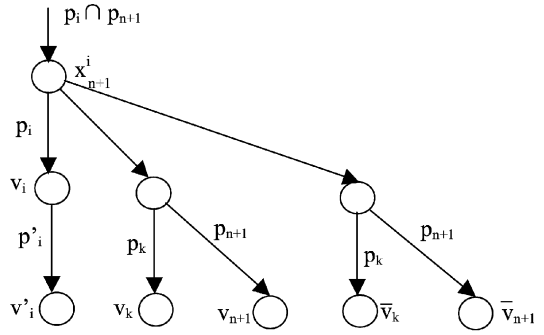

Fig. A.3. XML tree illustrating the proof of Axiom A5.

To establish (a.42) for the case of (C), the same arguments used for case (B) apply since the node $\bar{v}_i$ is chosen the same way in case (C) as in case (B).

Consider next (a.43) and the first case where $v' = v'_{n+1}$, and so we need to establish that $closest(\bar{v}_i, v'_{n+1})$ is true. Since $v_i = v'_i$ from (a.39), and $closest(v_i, v_{n+1})$ is true and $closest(v'_i, v'_{n+1})$ is true from (b), and $p_i \succ p_{n+1}$ from (C), then it follows from Lemma 20 (iv) that

$$v_i \in aancestor(v_{n+1}) \quad \text{and} \quad v_i \in aancestor\big(v'_{n+1}\big). \tag{a.79}$$

First, we claim that

$$p'_i \cap p_i = p'_i \cap p_{n+1}. \tag{a.80}$$

To show this, since $p_i \succ p_{n+1}$ from (C), then taking the intersection of both sides with p'_i and applying Lemma 19 (ix) we deduce that $p'_i \cap p_i \succeq p'_i \cap p_{n+1}$. For the reverse direction, since $p'_i \cap p_{n+1} \succeq p_i$ from (i) of Axiom A5, then taking the intersection of both sides with p'_i and applying Lemma 19 (ix) we obtain that $p'_i \cap p_{n+1} \succeq p'_i \cap p_i$ from and so (a.80) holds as claimed.

Next, since $closest(\bar{v}_i, v_i)$ is true from (a.42), there exists a node $x^i_i \in [\![p'_i \cap p_i]\!]$ such that $x^i_i \in aancestor(v_i)$ and $x^i_i \in aancestor(\bar{v}_i)$. Combining this with the fact that $v_i \in aancestor(v'_{n+1})$ from (a.79), we deduce that $x^i_i \in aancestor(v'_{n+1})$. Hence $closest(\bar{v}_i, v'_{n+1})$ is true as required because $p'_i \cap p_i = p'_i \cap p_{n+1}$ and thus $x^i_i \in [\![p'_i \cap p_{n+1}]\!]$.

Consider next the case where $v' \neq v'_{n+1}$. Then $v' = v'_k$ for some node v'_k in the set $v'_1, \ldots, v'_n$, or $v' = \bar{v}'_k$ some node $\bar{v}'_k$ in the set $\bar{v}'_1, \ldots, \bar{v}'_{i-1}$. So to establish (a.43), since $\bar{v}'_i = \bar{v}_i$ from (a.39), we need to show that

$$closest\big(\bar{v}_i, v'_k\big) \text{ is true for arbitrary } v'_k \text{ in } v'_1, \ldots, v'_n \tag{a.81}$$

and

$$closest\big(\bar{v}_i, \bar{v}'_k\big) \text{ is true for arbitrary } \bar{v}'_k \text{ in } \bar{v}'_1, \ldots, \bar{v}'_{i-1} \tag{a.82}$$

Consider (a.81). If $v'_k = v_k$, then the result is immediate from (a.42), so assume that $v'_k \neq v_k$. In this case, since $p_i \succ p_{n+1}$ from (C), it follows from Lemma 19 (iv) that $p_i \cap p_{n+1} = p_i$ and so (a.37) becomes

$$p_i \succ p_k \cap p_{n+1}. \tag{a.83}$$

Next, we claim that

$$p'_i \cap p_k = p'_i \cap p_{n+1}. \tag{a.84}$$

To show this, it follows from Lemma 19 (iii) that

$$p'_i = \big(p'_i \cap p_{n+1}\big).z_1 \tag{a.85}$$

and

$$p_{n+1} = \big(p'_i \cap p_{n+1}\big).z_2 \tag{a.86}$$

where

$$z_1 \cap z_2 = \epsilon. \tag{a.87}$$

Using Lemma 19 (iii) again, we also deduce that

$$p_k = (p_k \cap p_{n+1}) * z_3 \tag{a.88}$$

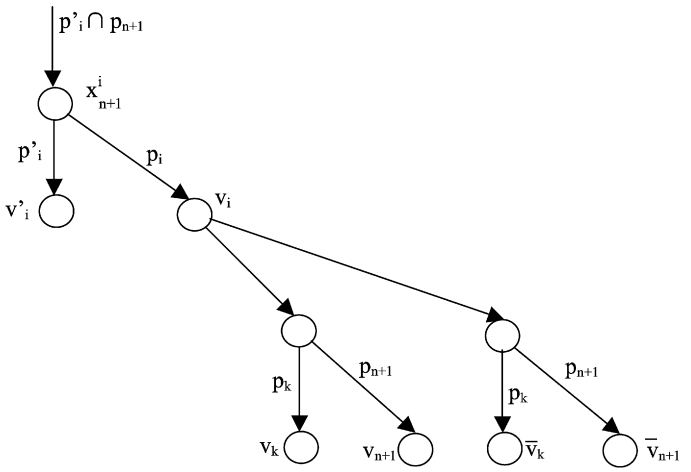

Fig. A.4. XML tree Illustrating the proof of Axiom A5.

and

$$p_{n+1} = (p_k \cap p_{n+1}) * z_4 \tag{a.89}$$

where

$$z_3 \cap z_4 = \epsilon. \tag{a.90}$$

From (a.83) and Lemma 19 (iii) and (iv), we find that $p_k \cap p_{n+1} = p_i * z_5$, where $z_5 \neq \epsilon$, and since $p'_i \cap p_{n+1} \succeq p_i$ from (i) of Axiom A5, then $p_i = (p'_i \cap p_{n+1}) * z_6$. Combining these two intermediate results and substituting into (a.88) and (a.89) we obtain

$$p_k = \big(p'_i \cap p_{n+1}\big).z_6.z_5.z_3 \tag{a.91}$$

and

$$p_{n+1} = \big(p'_i \cap p_{n+1}\big).z_6.z_5.z_4. \tag{a.92}$$

Combining then (a.85) and (a.91) we obtain

$$p'_i \cap p_k = \big(\big(p'_i \cap p_{n+1}\big).z_1\big) \cap \big(p'_i \cap p_{n+1}\big).z_6.z_5.z_3. \tag{a.93}$$

However, comparing (a.86) and (a.92), we deduce that $z_6.z_5.z_4 = z_2$, and so from (a.87) we find that $(z_6.z_5.z_4) \cap z_1 = \epsilon$. If $z_1 = \epsilon$ then (a.93) reduces to (a.84) as claimed. If instead $z_1 \neq \epsilon$ and $z_6 \neq \epsilon$, then z_1 and z_6 must start with different labels and so $(z_6.z_5.z_3) \cap z_1 = \epsilon$, and thus (a.93) again reduces to (a.84) as claimed. If instead $z_1 \neq \epsilon$ and $z_6 = \epsilon$, then since $(z_6.z_5.z_4) \cap z_1 = \epsilon$ and $z_5 \neq \epsilon$, then z_1 and z_5 must start with different labels and so $(z_6.z_5.z_3) \cap z_1 = \epsilon$ and thus (a.93) again reduces to (a.84) as claimed.

Next, since we have already established that $closest(\bar{v}_i, v'_{n+1})$ is true, it follows that there exists a node $x^i_{n+1} \in [\![p'_i \cap p_{n+1}]\!]$ such that $x^i_{n+1} \in aancestor(\bar{v}_i)$ and $x^i_{n+1} \in aancestor(v'_{n+1})$. Also, since $closest(v'_i, v'_{n+1})$ is true from (b) and $v_i = v'_i$ from (a.39), then $closest(v_i, v'_{n+1})$ is true. Combining this with $p_i \succ p_{n+1}$ from (C) and applying Lemma 20 (iv) shows that $v_i \in aancestor(v'_{n+1})$. So since x^i_{n+1} and v_i are both *aancestors* of v'_{n+1} and $p'_i \cap p_{n+1} \succeq p_i$ from (i) of Axiom A5, then $x^i_{n+1} \in aancestor(v_i)$ from Lemma 20 (iv). Also, since $closest(v'_k, v'_{n+1})$ is true from (b), there exists a node $\bar{x}^k_{n+1} \in [\![p_k \cap p_{n+1}]\!]$ such that $\bar{x}^k_{n+1} \in aancestor(v'_k)$ and $\bar{x}^k_{n+1} \in aancestor(v'_{n+1})$. Combining this with the fact that $p_i \succ p_k \cap p_{n+1}$ from (a.83) and $v_i \in aancestor(v'_{n+1})$, we deduce that $v_i \in aancestor(\bar{x}^k_{n+1})$ and hence $v_i \in aancestor(v'_k)$. Combining this then with $x^i_{n+1} \in aancestor(v_i)$, we deduce that $x^i_{n+1} \in aancestor(v'_k)$. It follows that $closest(\bar{v}_i, v'_k)$ is true since $x^i_{n+1} \in [\![p'_i \cap p_k]\!]$ because $p'_i \cap p_k = p'_i \cap p_{n+1}$ from (a.84), and thus (a.81) is established. The situation is illustrated in Fig. A.4.

Finally, to establish (a.82), the same arguments used to establish (a.47) for case (B) apply.

Proof of Lemma 36. (i) We can define the size of the tree $\mathbb{T}$, denoted by $|\mathbb{T}|$, to be the number of nodes in $\mathbb{T}$ plus the sum of all text values in $\mathbb{T}$. At each iteration of the outer loop, a rule is applied either to a node with an element label or to a non-element node. If it is an element node, then $|\mathbb{T}|$ strictly decreases because at least one node is deleted from $\bar{\mathbb{T}}$, and if it is a non-element node then $|\mathbb{T}|$ strictly decreases since the *val* of a node is replaced by a strictly smaller *val*. Hence at every iteration, $|\mathbb{T}|$ strictly decreases and so Algorithm 2 must terminate because $|\mathbb{T}|$ is bounded from below by 0.

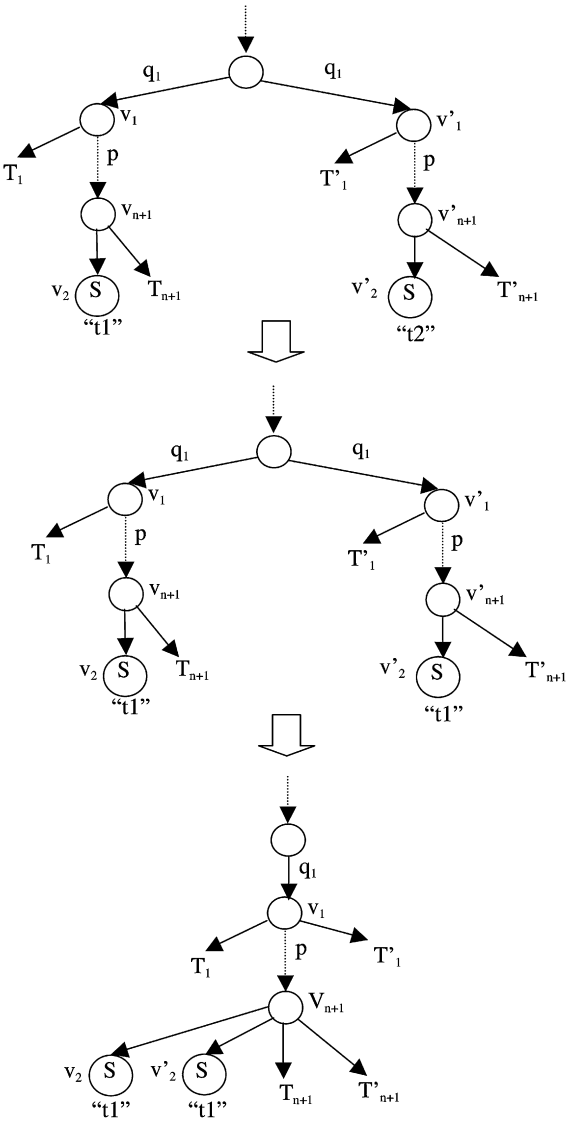

Fig. A.5.

(ii) The result is by induction on the number of subtree mergers. Initially the result is true because of the restriction placed on the input tree $\mathbb{T}$ by Algorithm 2. Assume then that it is true after the merging of n subtrees and let $\mathbb{T}_n$ be the resulting tree. Then $\mathbb{T}_{n+1}$ must also be complete or else it contradicts the fact that $\mathbb{T}_n$ is complete.

(iii) From the definition of the algorithm, the algorithm terminates only when there is no XFD that is violated.

(iv) Let us denote by φ_1 the operations performed in the chase if the test at Line 5 of Algorithm 2 is satisfied, and let φ_2 denote the operations performed in the chase when the test at Line 11 is satisfied. Hence $\mathbb{T} \overset{\varphi_1}{\Rightarrow} \mathbb{T}_1$ denotes the fact that $\mathbb{T}$ is transformed to $\mathbb{T}_1$ by applying φ_1, and $\mathbb{T} \overset{\varphi_2}{\Rightarrow} \mathbb{T}_2$ denotes the fact that $\mathbb{T}$ is transformed to $\mathbb{T}_2$ by applying φ_2. Also, let $\mathbb{T} \overset{*}{\Rightarrow} \mathbb{T}_1$ denote that $\mathbb{T}$ is transformed into $\mathbb{T}_1$ by a sequence of either φ_1 or φ_2 operations.

Since we have shown that the chase always terminates, from Theorem 8.8 in [23] it suffices to show the following: if $\mathbb{T} \overset{\varphi_1}{\Rightarrow} \mathbb{T}_1$ and $\mathbb{T} \overset{\varphi_2}{\Rightarrow} \mathbb{T}'_1$, then there is an XML tree $\mathbb{T}_2$ such that $\mathbb{T}_1 \overset{*}{\Rightarrow} \mathbb{T}_2$ and $\mathbb{T}'_1 \overset{*}{\Rightarrow} \mathbb{T}_2$. We will do this by showing that if $\mathbb{T} \overset{\varphi_1}{\Rightarrow} \overset{\varphi_2}{\Rightarrow} \mathbb{T}_1$ and $\mathbb{T} \overset{\varphi_2}{\Rightarrow} \overset{\varphi_1}{\Rightarrow} \mathbb{T}'_1$, then $\mathbb{T}_1 = \mathbb{T}'_1$. Suppose first that φ_2 is a result of applying an XFD which has a path p_{n+1} on the r.h.s. to nodes v_{n+1} and v'_{n+1}, where $Last(p_{n+1}) \in \mathbf{E}$ and v_{n+1} and v'_{n+1} are in $[\![p_{n+1}]\!]$. Assume also that the application of φ_2 then involves having to merge the pairs of subtrees rooted at v_{n+1} and v'_{n+1}, through to the pair of subtrees that are rooted at v_1 and v'_1.

For φ_1, there are two possibilities: the path on the r.h.s. of the XFD ends in a text label, or the path ends in an attribute label. For the first case, we assume that p_{n+1} is of the form $p * z$, where $z = \mathtt{S}$. The result of the transformation $\mathbb{T} \overset{\varphi_1}{\Rightarrow} \overset{\varphi_2}{\Rightarrow} \mathbb{T}_1$ is shown in Fig. A.5.

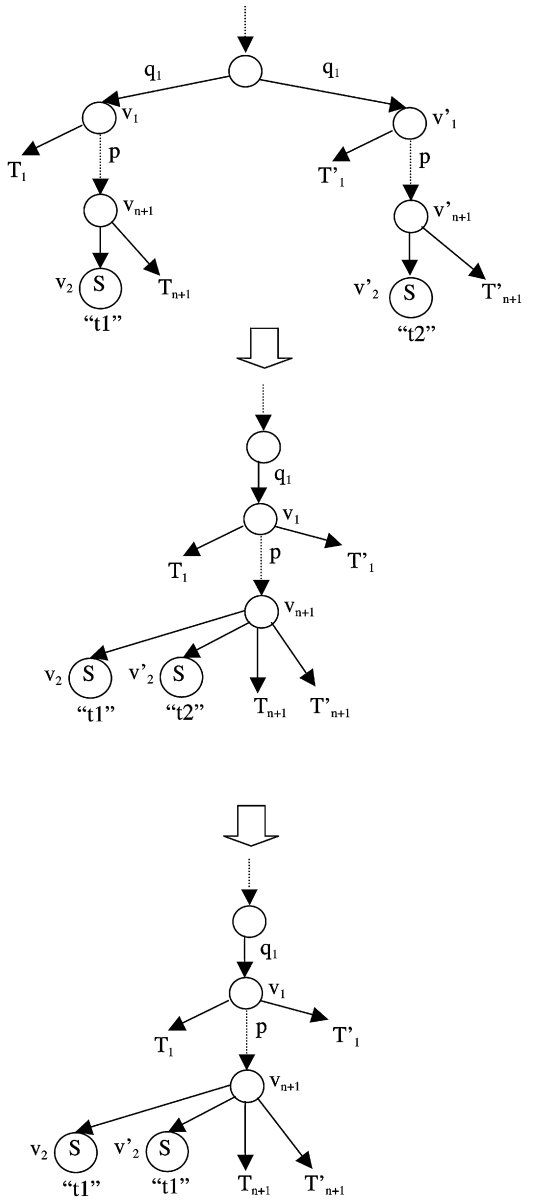

Fig. A.6.

If we change the order of the operations, then result of $\mathbb{T} \overset{\varphi_2}{\Rightarrow} \overset{\varphi_1}{\Rightarrow} \mathbb{T}_1$ is shown in Fig. A.6.

As can be seen from these figures, the final tree is the same in both cases, irrespective of the ordering of the operations.

The second case to consider is where the application of φ_1 results from using an XFD where p_{n+1} is of the form $p_i * z$, where $z = @l$. The result of the transformation $\mathbb{T} \overset{\varphi_1}{\Rightarrow} \overset{\varphi_2}{\Rightarrow} \mathbb{T}_1$ is shown in Fig. A.7.

If we change the order of the operations, then the result of $\mathbb{T} \overset{\varphi_2}{\Rightarrow} \overset{\varphi_1}{\Rightarrow} \mathbb{T}_1$ is shown in Fig. A.8.

As can be seen from these figures, the final tree again is the same in both cases, irrespective of the ordering of the operations. So in all cases we can interchange the order of operations, and so the chase satisfies the Church–Rosser property.

(v) Let $\mathbb{T}_0 = \mathit{Chase}(\mathbb{T}, \mathbb{P}, \Sigma)$, $\mathbb{T}'_0 = \mathit{Chase}(\mathbb{T}, \mathbb{P}, \Sigma')$, $\mathbb{T}_1 = \mathit{Chase}(\mathbb{T}_0, \mathbb{P}, \Sigma')$ and $\mathbb{T}'_1 = \mathit{Chase}(\mathbb{T}'_0, \mathbb{P}, \Sigma)$. From (iii) we derive that $\mathbb{T}_0 \models \Sigma$ and $\mathbb{T}'_0 \models \Sigma'$, and since $\Sigma \equiv \Sigma'$ this implies that $\mathbb{T}_0 \models \Sigma'$ and $\mathbb{T}'_0 \models \Sigma$. It follows that the chase does not modify either $\mathbb{T}_0$ or $\mathbb{T}_1$ and so $\mathbb{T}_1 = \mathbb{T}_0$ and $\mathbb{T}'_1 = \mathbb{T}'_0$. However, we derive from (iv) that $\mathbb{T}_1 = \mathbb{T}'_1$ and hence $\mathbb{T}_0 = \mathbb{T}'_0$ as claimed.

We now present the proofs of the remaining results of Section 5, starting again with some preliminary lemmas. Also, to simplify the notation in the proofs, we will denote $\mathit{Chase}(\mathbb{T}_P, \mathbb{P}, \underset{P}{\Sigma})$ by $\mathbb{T}$. The next four results are immediate consequences of the definitions of Algorithms 2 and 3 and the proofs are omitted.

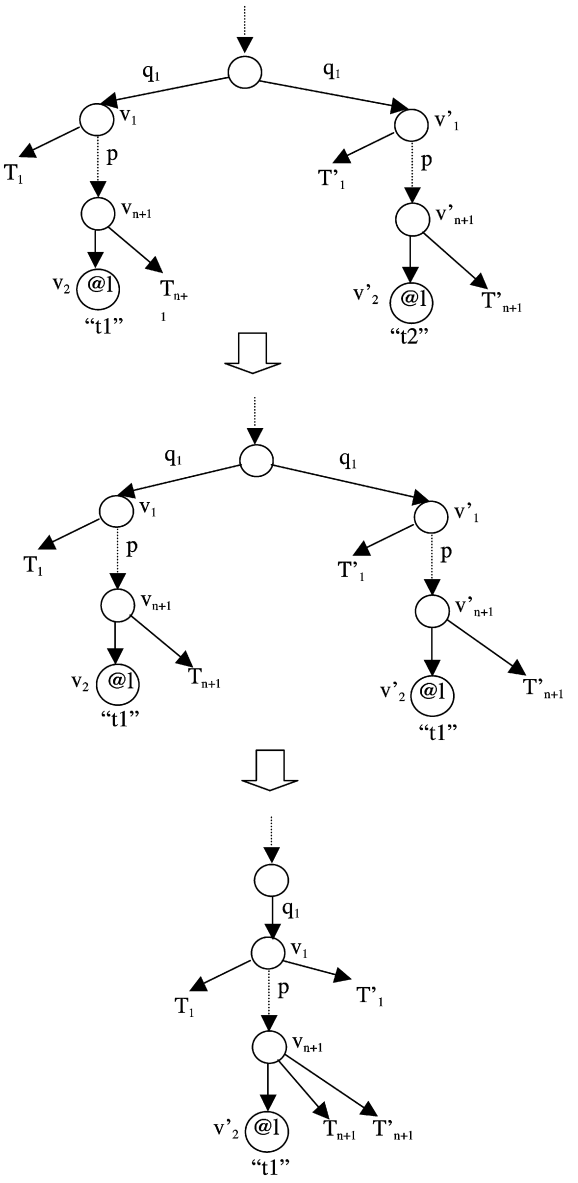

Fig. A.7.

Lemma 55. *If $p \in \mathbb{P}$, then $|[\![p]\!]| = 1$ in $\mathbb{T}_P$ iff $p = \rho$ or ($\rho = Parent(p)$ and $Last(p) \in \mathbf{A}$), else $|[\![p]\!]| = 2$.*

Lemma 56. *If $p \in \mathbb{P}$ and $Last(p) \in \mathbf{E} \cup \mathbf{A}$, then $|[\![p]\!]| \leqslant 2$ in $\mathbb{T}$.*

Lemma 57. *If $p \in \mathbb{P}$ and $Last(p) = \mathcal{S}$, then $|[\![p]\!]| = 2$ in $\mathbb{T}$.*

Lemma 58. *If $p_i \in P$ and $[\![p_i]\!] = \{v, v'\}$ in $\mathbb{T}$, then $val(v) = val(v')$.*

Lemma 59. *Let $p \in \mathbb{P}$ and let v and v' be nodes in $[\![p]\!]$ in $\mathbb{T}_P$ such that $v \neq v'$. If there exists a node $\bar{v} \in \mathbb{T}_P$ such that $\bar{v} \in aancestor(v)$ and $\bar{v} \in aancestor(v')$, then $\bar{v} = v_\rho$, where v_ρ is the root node of $\mathbb{T}_P$.*

Proof. Suppose to the contrary that node $\bar{v} \neq v_\rho$. Then $\bar{v} \in [\![s]\!]$ for some path s such that, by Lemma 20 (x), $s \succeq p$ and $s \neq \rho$. So by Lemma 55, there exists another node $\bar{v}' \in [\![s]\!]$, in $\mathbb{T}_P$ such that $\bar{v} \neq \bar{v}'$. Since $\mathbb{T}_P$ is complete and $s \succeq p$, there exists a node $v'' \in [\![p]\!]$ such that $\bar{v}' \in aancestor(v'')$. However, because $\bar{v} \neq \bar{v}'$ and $\mathbb{T}_P$ is a tree, v'' must be distinct from both v and v', which contradicts Lemma 55, and so we conclude that $\bar{v} = v_\rho$. □

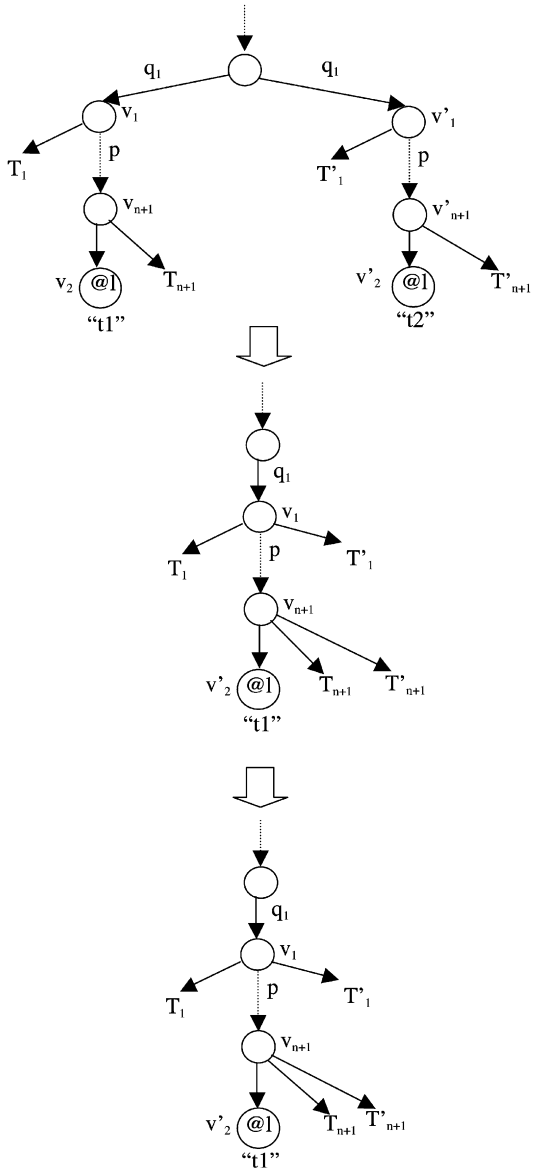

Fig. A.8.

Lemma 60. *If the following hold*:

(i) $q \in \mathbb{P}$;
(ii) *there exist nodes* $v, v' \in [\![q]\!]$ *in* $\mathbb{T}$ *such that* $v \neq v'$;
(iii) *there exists a node* $\bar{v}$ *and a path* s *such that* $\bar{v} \in [\![s]\!]$, $\bar{v} \neq v_\rho$, *and* $\bar{v} \in aancestor(v)$ *and* $\bar{v} \in aancestor(v')$;

then $s \in clo(P)$ *and* $s \succeq q$.

Proof. We prove the result by induction on the number of steps in constructing $\mathbb{T}$ by Algorithm 2. Initially, by Lemma 59, such nodes v and v' do not exist in $\mathbb{T}_P$ and so the inductive hypothesis trivially holds. Assume then that the inductive hypothesis holds after iteration $k-1$ of the outer repeat loop in Algorithm 2. By definition of Algorithm 2, if there are nodes v and v' that have a common non-root ancestor $\bar{v}$ after iteration k, then an ancestor node of v and an ancestor node of v' in $[\![s]\!]$, for some path s, must have been merged during iteration k. For this to happen, by definition of Algorithm 2 and the fact that we are using $clo(P)$ as input to Algorithm 2, this must have occurred from applying an XFD $p'_1, \ldots, p'_n \to s'$, where $\{p'_1, \ldots, p'_n\} \subseteq clo(P)$ and $s \succeq s' \succ q$. However, since $\{p'_1, \ldots, p'_n\} \subseteq clo(P)$, then $s' \in clo(P)$ from Line 13 (b) of Algorithm 1, and then, from Lines 14–19 of Algorithm 1, $s \in clo(P)$ as required. □

Lemma 61. *If $s \in \mathbb{P}$ and there exist v_s, v'_s in $[\![s]\!]$ in $\mathbb{T}$ such that $val(v_s) \neq val(v'_s)$, then $\mathbb{T} \not\models P \rightarrow s$.*

Proof. Let $q = p_1 \cap s$ (remembering that $P = \{p_1, \ldots, p_n\}$). Let v_q be the node in $[\![q]\!]$ in $\mathbb{T}$ such that $v_q \in aancestor(v_s)$ and let v_{p_1} be any node in $[\![p_1]\!]$ that is a descendant of v_q (such a node exists because $q \succeq p_1$ from Lemma 19 (vii) and $\mathbb{T}$ is complete). It is immediate from this construction that $closest(v_{p_1}, v_s)$ is true. Then by repeatedly applying Lemma 21 we can extend the set $\{v_{p_1}, v_s\}$ to the set $\{v_{p_1}, \ldots, v_{p_n}, v_s\}$ such that *closest* is true for every pair of nodes in $\{v_{p_1}, \ldots, v_{p_n}, v_s\}$.

Similarly, if v'_q is any node in $[\![q]\!]$ such that $v'_q \in aancestor(v'_s)$, then there exists a node $v'_{p_1} \in [\![p_1]\!]$ that is a descendant of v'_q and $closest(v'_{p_1}, v'_s)$ is true. Again, by repeatedly applying Lemma 21, we can extend the set $\{v'_{p_1}, v'_s\}$ to the set $\{v'_{p_1}, \ldots, v'_{p_n}, v'_s\}$ such that *closest* is true for every pair of nodes in $\{v'_{p_1}, \ldots, v'_{p_n}, v'_s\}$. Then by Lemma 58, for all $i \in [1, n]$, $val(v_{p_i}) = val(v'_{p_i})$ but $val(v_s) \neq val(v'_s)$ by assumption and so $\mathbb{T} \not\models P \rightarrow s$. □

Lemma 62. *If $q \in \mathbb{P}$ and $Last(q) = S$, then $q \in clo(P)$ iff there exist $v_1, v_2 \in [\![q]\!]$ in $\mathbb{T}$ such that $v_1 \neq v_2$ and $val(v_1) = val(v_2)$.*

Proof. *If*: Suppose first that the *val*'s of both nodes v_1 and v_2 are the same in $\mathbb{T}$ and $\mathbb{T}_P$. Then from the definition of Algorithm 3, this means that $q \in P$ and so $q \in clo(P)$ from Line 1 of Algorithm 1. If instead the *val* of either v_1 or v_2 is changed during the execution of Algorithm 2, then this can only happen if either Line 8 or Line 9 is executed. Remembering that we are using $\sum_P$ as input to Algorithm 2, this means that there is a $p_{n+1} \in clo(P)$ such that $q = p_{n+1}$, which establishes the result.

Only If: By Lemma 57, there exists a pair of distinct nodes $v_1, v_2 \in [\![q]\!]$ and so we shall show the contrapositive that if $val(v_1) \neq val(v_2)$, then $q \notin clo(P)$. If $val(v_1) \neq val(v_2)$, then $\mathbb{T} \not\models P \rightarrow q$ from Lemma 61, and so, since $\mathbb{T} \models \sum_P$ by Lemma 36 (iii), $q \notin clo(P)$. □

Lemma 63. *If $q \in \mathbb{P}$ and $Last(q) \in \mathbf{E}$, then $q \in clo(P)$ iff $|[\![q]\!]| = 1$ in $\mathbb{T}$.*

Proof. *If*: If $q = \rho$, then it follows from Line 1 of Algorithm 1 that $\rho \in clo(P)$, so suppose instead that $q \neq \rho$. In this case, from Algorithm 3, $|[\![q]\!]| = 2$ in $\mathbb{T}_P$. So from the definition of Algorithm 2, this means that Line 12 must have been executed and so there exists XFD $P \rightarrow p_{n+1}$ from $\sum_P$, where $q \succeq p_{n+1}$ and $Last(p_{n+1}) \in \mathbf{E}$. However, by Line 14 and Line 19 in Algorithm 1 this implies that $q \in clo(P)$ as required.

Only If: We show the contrapositive that if $|[\![q]\!]| \neq 1$ in $\mathbb{T}$ then $q \notin clo(P)$. If $|[\![q]\!]| \neq 1$ in $\mathbb{T}$, then the *val* of the two nodes in $[\![q]\!]$ must be different and so from Lemma 61 it follows that $\mathbb{T} \not\models P \rightarrow q$. This implies that $q \notin clo(P)$ since $\mathbb{T} \models \sum_P$ by Lemma 36 (iii). □

Lemma 64. *If $q \in \mathbb{P}$ and $Last(q) \in \mathbf{A}$, then $q \in clo(P)$ iff $|[\![q]\!]| = 1$ in $\mathbb{T}$ or $val(v) = val(v')$, where $\{v, v'\} \subseteq [\![q]\!]$ in $\mathbb{T}$.*

Proof. *If*: If $val(v) = val(v')$ then the same proof as in Lemma 62 applies, and so we assume that $|[\![q]\!]| = 1$ in $\mathbb{T}$. There are then two possibilities: (a) $|[\![q]\!]| = 1$ in $\mathbb{T}_P$; (b) $|[\![q]\!]| \neq 1$ in $\mathbb{T}_P$. If (a) holds, then from the definition of Algorithm 2 we must have that $Parent(q) = \rho$ and so $q \in clo(P)$ since $\rho \in clo(P)$ from Line 1 of Algorithm 1 and hence $q \in clo(P)$ from Line 9. If instead (b) holds, one of the q nodes must have been deleted during the execution of Algorithm 2. Therefore, there has to be a path q' in $\mathbb{P}$ such that $Parent(q) = q'$ and a path $s \in \mathbb{P}$ such that $q' \succeq s$ and an XFD $P \rightarrow s \in \sum_P$. It follows then from Lines 5 and 20 of Algorithm 1 that $q' \in clo(P)$, and so $q \in clo(P)$ from Lines 9 and 18 of Algorithm 1.

Only If: The proof is by contra-positive and so assume $|[\![q]\!]| \neq 1$ and $val(v) \neq val(v')$ in $\mathbb{T}$. It follows from Lemma 61 that $\mathbb{T} \not\models P \rightarrow q$, and so $q \notin clo(P)$ since $\mathbb{T} \models \sum_P$ by Lemma 36 (iii). □

Proof of Theorem 37 (i). The proof is by contra-positive and so assume that $T \not\models \sigma$, where σ is an XFD $r_1, \ldots, r_k \rightarrow r_{k+1}$ in Σ. Then there exist nodes $v_1 \in [\![r_1]\!], \ldots, v_k \in [\![r_k]\!], v_{k+1} \in [\![r_{k+1}]\!]$ and $v'_1 \in [\![r_1]\!], \ldots, v'_k \in [\![r_k]\!], v'_{k+1} \in [\![r_{k+1}]\!]$ in $\mathbb{T}$ such that

$$\text{for all } i, j \in [1, k+1], \quad closest(v_i, v_j) = closest\big(v'_i, v'_j\big) = \text{true} \tag{a.94}$$

and

$$val(v_{k+1}) \neq val\big(v'_{k+1}\big) \quad \text{and for all } i \in [1, k], val(v_i) = val\big(v'_i\big). \tag{a.95}$$

From (a.94) and the definition of *closest*, for all $i \in [1, k]$ there exist nodes x^i_{k+1} and x'^i_{k+1} in $\mathbb{T}$ such that

$$\begin{aligned} &x^i_{k+1} \in [\![r_i \cap r_{k+1}]\!] \quad \text{and} \quad x^i_{k+1} \in aancestor(v_i) \\ &\text{and} \quad x^i_{k+1} \in aancestor(v_{k+1}) \end{aligned} \tag{a.96}$$

and

$$x'^{i}_{k+1} \in [\![r_i \cap r_{k+1}]\!] \quad \text{and} \quad x'^{i}_{k+1} \in \mathit{aancestor}(v'_i) \quad \text{and} \quad x'^{i}_{k+1} \in \mathit{aancestor}(v'_{k+1}). \tag{a.97}$$

We consider three exhaustive cases:

(a) $x^i_{k+1} = x'^{i}_{k+1} \neq v_\rho$;
(b) $x^i_{k+1} = x'^{i}_{k+1} = v_\rho$;
(c) $x^i_{k+1} \neq x'^{i}_{k+1}$.

For each of (a)–(c), and for each $i \in [1,k]$, we choose a path $q_i \in \mathbb{P}$ as follows. If (a) holds, we let $q_i = r_i \cap r_{k+1}$; if (b) holds we let $q_i = \rho$; if (c) holds we let $q_i = r_i$. We claim that $q_i \in \mathit{clo}(P)$ in all cases.

To show this, assume first that (a) holds. Since $\mathit{val}(v_{k+1}) \neq \mathit{val}(v'_{k+1})$ from (a.95), it follows that $v_{k+1} \neq v'_{k+1}$ and so since $x^i_{k+1} = x'^{i}_{k+1} \neq v_\rho$ from (a), v_{k+1} and v'_{k+1} have a common ancestor, $x^i_{k+1} \in [\![r_i \cap r_{k+1}]\!]$, that is not v_ρ. So by Lemma 60, $r_i \cap r_{k+1} \in \mathit{clo}(P)$ and so $q_i \in \mathit{clo}(P)$. If (b) occurs, then since $x^i_{k+1} = v_\rho$ and $x^i_{k+1} \in [\![r_i \cap r_{k+1}]\!]$ by (a.97), we have that $r_i \cap r_{k+1} = \rho$, and $\rho \in \mathit{clo}(P)$ from Line 1 of Algorithm 1. Hence $q_i \in \mathit{clo}(P)$. If (c) occurs, then since $x^i_{k+1} \neq x'^{i}_{k+1}$ and x^i_{k+1} is an ancestor of v_i, and x'^{i}_{k+1} is an ancestor of v'_i, it follows that $v_i \neq v'_i$ since $\mathbb{T}$ is a tree. However, since $\mathit{Last}(r_i) \notin \mathbf{E}$ because Σ is a text set and $\mathit{val}(v_i) = \mathit{val}(v'_i)$ from (a.95), applying Lemma 62 in the case that $\mathit{Last}(r_i) = \mathtt{S}$, or Lemma 64 in the case that $\mathit{Last}(r_i) \in \mathbf{A}$, shows that $r_i \in \mathit{clo}(P)$ and so $q_i \in \mathit{clo}(P)$. Hence for all possible cases, $q_i \in \mathit{clo}(P)$ as claimed.

Next, we claim that the XFD $r_1, \ldots, r_k \to r_{k+1}$ and the paths $q_1, \ldots, q_k$ satisfy Line 13 (a) of Algorithm 1, and so r_{k+1} is added to $\mathit{clo}(P)$ at Line 14. To verify this, we consider the various cases that can occur for q_i. If (a) holds, then $q_i = r_i \cap r_{k+1}$ and so $(r_i \cap r_{k+1}) \succeq (r_i \cap r_{k+1})$ is immediate, and $(r_i \cap r_{k+1}) \succeq r_i$ follows from Lemma 19 (vii). If (b) holds, then $q_i = \rho = r_i \cap r_{k+1}$, and so $r_i \cap r_{k+1} \succeq q_i$ is immediately satisfied, and $q_i \succeq r_i$ is satisfied since ρ is a prefix of every path in $\mathbb{P}$. Finally, if (c) holds, then $q_i = r_i$ and so $r_i \cap r_{k+1} \succeq q_i$ follows from Lemma 19 (vii), and $q_i \succeq r_i$ follows immediately. Thus Line 13 (a) of Algorithm 1 is satisfied and so r_{k+1} is added to $\mathit{clo}(P)$ at Line 14. This implies, from Lemma 36 (iii), that $\mathbb{T} \models P \to r_{k+1}$ which is a contradiction since by applying Lemma 61 to (a.95), we obtain that $\mathbb{T} \not\models P \to r_{k+1}$. We conclude that $\mathbb{T} \models \sigma$, and thus $\mathbb{T} \models \Sigma$ as claimed. □

Proof of Theorem 37 (ii). If $q \notin \mathit{clo}(P)$ and $\mathit{Last}(q) \in \mathbf{E}$, then $|[\![q]\!]| \neq 1$ in $\mathbb{T}$ by Lemma 63 and thus the *val*'s of the two nodes in $[\![q]\!]$ are different. From Lemma 61, this implies that $\mathbb{T} \not\models P \to q$.

If $q \notin \mathit{clo}(p)$ and $\mathit{Last}(q) = \mathtt{S}$, applying Lemma 62 shows that $[\![q]\!] = \{v_1, v_2\}$ and $\mathit{val}(v_1) \neq \mathit{val}(v_2)$, for some nodes v_1 and v_2 in $\mathbb{T}$. Applying Lemma 61 then shows that $\mathbb{T} \not\models P \to q$.

Finally, if $q \notin \mathit{clo}(p)$ and $\mathit{Last}(q) \in \mathbf{A}$, then using Lemma 64 shows that $[\![q]\!] = \{v_1, v_2\}$ and $\mathit{val}(v_1) \neq \mathit{val}(v_2)$, for some nodes v_1 and v_2 in $\mathbb{T}$. Again, applying Lemma 61 shows that $\mathbb{T} \not\models P \to q$. □

Proof of Corollary 42. (i) ⇒ (ii) follows since by the theorem, if $(\mathbb{P}, \Sigma) \vdash \sigma$, then $(\alpha(\mathbb{P}), \alpha(\Sigma)) \vdash \alpha(\sigma)$, and then $\mathit{Chase}(\alpha(\mathbb{T}_P))$, $\alpha(\mathbb{P}), \alpha(\Sigma) \models \sigma$ follows from Corollary 40.

For (ii) ⇒ (iii), it follows from Corollary 40 and Theorem 38 that if (ii) holds then $\alpha(q) \in \mathit{clo}(\alpha(P))$. We now show that if $\alpha(q) \in \mathit{clo}(\alpha(P))$ then $q \in \mathit{clo}(P)$. The proof is by induction on the number of iterations in the repeat loop of Algorithm 1. After Line 1, we have that $\mathit{clo}(\alpha(P)) = \{\alpha(p_1), \ldots, \alpha(p_n), \alpha(\rho)\}$ and $\mathit{clo}(P) = \{p_1, \ldots, p_n, \rho\}$ and so if $\alpha(q) \in \mathit{clo}(\alpha(P))$ then $q \in \mathit{clo}(P)$. At Line 5, if a path $\alpha(q)$ is added to $\mathit{clo}(\alpha(P))$ then we must have, since $\alpha(q) \succ \alpha(p)$ for some $p \in \{p_1, \ldots, p_n, \rho\}$, that $\mathit{Last}(\alpha(q)) \in \mathbf{E}$. Hence $\alpha(q) = q$ from the definition of α, and so if $\alpha(q) \succ \alpha(p)$ then $q \succ \alpha(p)$ and thus either $q \succ p$ or $q = p$. Hence, noting that $p \in \mathit{clo}(P)$ from Line 1, if $q \succ p$ then $q \in \mathit{clo}(P)$ from Line 5 and if $q = p$ then q is already in $\mathit{clo}(P)$ from Line 1. Next, if $\alpha(q')$ is added at Line 9, then there must exist $\alpha(p) \in \mathit{clo}(\alpha(P))$ such that $\alpha(p) = \mathit{Parent}(\alpha(q'))$. Since $\alpha(p) = \mathit{Parent}(\alpha(q'))$, $\mathit{Last}(\alpha(p)) \in \mathbf{E}$ and so $\alpha(p) = p$. There are then two possibilities. If $\alpha(q') = q$ then q must be added to $\mathit{clo}(P)$ at Line 9. If instead $\alpha(q') = q * l_@$, then $q = p$ and $p \in \mathit{clo}(P)$ since $\alpha(p) \in \mathit{clo}(\alpha(P))$, so q must be added to $\mathit{clo}(P)$ at Line 9.

So at the start of the repeat loop, $\alpha(q) \in \mathit{clo}(\alpha(P))$ and so assume inductively that the property holds after m iterations of the repeat loop and consider iteration $m+1$. Suppose firstly that Line 13 (a) evaluates to true, and so $\alpha(r_{k+1})$ is added to $\mathit{clo}(\alpha(P))$ at Line 14. For this to happen, we must have that $\alpha(r_i) \cap \alpha(r_{k+1}) \succeq \alpha(q_i)$ and $(\alpha(q_i) \succeq \alpha(r_i)$ or $\alpha(q_i) \succeq \alpha(r_{k+1}))$. Applying Lemma 32 (iv) shows that if $\alpha(r_i) \cap \alpha(r_{k+1}) \succeq \alpha(q_i)$ then $r_i \cap r_{k+1} \subseteq q_i$, and Lemma 32 (i) shows that if $\alpha(q_i) \subseteq \alpha(r_i)$ then $q_i \subseteq r_i$ and if $\alpha(q_i) \subseteq \alpha(r_{k+1})$ then $q_i \subseteq r_{k+1}$. So Line 13 (a) is satisfied and thus r_{k+1} is added to $\mathit{clo}(P)$ as required. Next, suppose that Line 13 (b) evaluates to true and so $\alpha(r_i) \cap \alpha(r_{k+1}) = \alpha(\rho)$ or $\alpha(r_i) \in \mathit{clo}(\alpha(P))$. If $\alpha(r_i) \cap \alpha(r_{k+1}) = \alpha(\rho)$, then $\alpha(r_i) \neq \alpha(r_{k+1})$ since Σ contains no redundant XFDs and so using Lemma 32 (iii) shows that $r_i \cap r_{k+1} = \rho$ as required. Alternatively, if $\alpha(r_i) \in \mathit{clo}(\alpha(P))$ then it follows from the inductive hypothesis that $r_i \in \mathit{clo}(P)$ and hence Line 13 (b) is satisfied and r_{k+1} will be added to $\mathit{clo}(P)$ at Line 14. Finally, if q is added at Line 18 or Line 20 then the inductive hypothesis and the same arguments used for the tests at Line 5 and Line 9 show that $q \in \mathit{clo}(P)$. Hence the inductive hypothesis holds and so (ii) ⇒ (iii) also holds.

Finally, (iii) ⇒ (i) follows from Lemma 34. □

Symbol	Meaning	Definition
$\mathbf{E}$	set of element names	Definition 1
$\mathbf{A}$	set of attribute names	Definition 1
$\mathtt{S}$	PCDATA	Definition 1
$\mathbb{T}$	an XML tree	Definition 1
V	set of nodes	Definition 1
v	node	Definition 1
v_ρ	root node in $\mathbb{T}$	Definition 1
l	label-sequence	Definition 3
p	path	Definition 4
$\tilde{v}$	path instance	Definition 5
$\succeq$	prefix relationship between paths or path instances	Definitions 4 and 5
$\cap$	intersection of paths or path instances	Definitions 4 and 5
$\mathbb{P}$	set of paths	Definition 4
$\mathbb{T} \models \mathbb{P}$	$\mathbb{T}$ conforms to $\mathbb{P}$	Definition 7
$[\![p]\!]$	the set of nodes reachable from the root via path p	Definition 8
$aancestor(v)$	either v or an ancestor of v	Definition 9
σ, Σ	an XFD and a set of XFDs	Definition 11
$\mathbb{T} \models \sigma$	$\mathbb{T}$ satisfies σ	Definition 14
$\mathbb{T} \models \Sigma$	$\mathbb{T}$ satisfies Σ	Definition 14
$\sigma \models \mathbb{P}$	σ is consistent with $\mathbb{P}$	Definition 14
$(\mathbb{P}, \Sigma) \vdash \sigma$	Σ logically implies σ	Definition 14
$(\mathbb{P}, \Sigma)^+$	the set of XFDs logically implied by Σ	Definition 14
$\equiv$	logical equivalence between sets of XFDs	Definition 14
$(\mathbb{P}, \Sigma, P)^+$	set of paths logically implied by a set of paths P	Definition 14
l_α	label not in $\mathbf{A}$	Definition 26
$ele(\mathbb{P})$	paths in $\mathbb{P}$ ending in an element label	Definition 26
$\alpha(p)$	conversion of a path p to a text path	Definition 26
$\alpha(\mathbb{P})$	conversion of paths in $\mathbb{P}$ to text paths	Definition 26
$\alpha(\sigma)$	text XFD	Definition 26
$\alpha(\Sigma)$	set of text XFDs	Definition 26
$\alpha(\mathbb{T})$	text XML tree	Definition 26
$\sqsubseteq$	subsumption	Definition 27
$(\mathbb{P}, \Sigma) \models^d \sigma$	Σ derives σ	Definition 18

Fig. A.9. Summary of notation.

References

[1] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison–Wesley, 1996.
[2] T. Bray, J. Paoli, C. Sperberg-McQueen, E. Maler, F. Yergeau, J. Cowan, Extensible markup language (XML) 1.1, w3C recommendation, http://www.w3.org/TR/xml111/, 2004.
[3] K. Beyer, R. Cochrane, V. Josifovski, J. Kleewein, G. Lapis, G. Lohman, System RX: One part relational, one part XML, in: ACM SIGMOD International Conference on the Management of Data, ACM, Paris, France, 2005, pp. 347–358.
[4] W. Fan, XML constraints: Specification, analysis, and applications, in: DEXA Workshops 2005, 2005, pp. 805–809.
[5] A. Halevy, A. Rajaraman, J.J. Ordille, Data integration: The teenage years, in: 32nd International Conference on Very Large Databases (VLDB), Morgan Kaufmann, Seoul, Korea, 2006, pp. 9–16.
[6] M. Arenas, L. Libkin, A normal form for XML documents, ACM Trans. Database Syst. 29 (1) (2004) 195–232.
[7] M. Arenas, L. Libkin, An information-theoretic approach to normal forms for relational and XML data, J. ACM 52 (2) (2005) 246–283.
[8] M. Arenas, Normalization theory for XML, SIGMOD Rec. 35 (4) (2006) 57–64.
[9] P. Atzeni, V. DeAntonellis, Relational Database Theory, Benjamin Cummings, 1993.
[10] M. Vincent, J. Liu, C. Liu, Strong functional dependencies and their application to normal forms in XML, ACM Trans. Database Syst. 29 (3) (2004) 445–462.
[11] P. Buneman, S. Davidson, W. Fan, C. Hara, W. Tan, Reasoning about keys for XML, Inf. Syst. 28 (8) (2003) 1037–1063.
[12] M. Vincent, J. Liu, M. Mohania, On the equivalence between FDs in XML and FDs in relations, Acta Inform. 44 (3) (2007).
[13] Ł. Kot, W. White, Characterization of the interaction of XML functional dependencies with DTDs, in: 11th International Conference on Database Theory (ICDT), Springer, Barcelona, Spain, 2007, pp. 119–133.
[14] N. Klarlund, T. Schwentick, D. Suciu, XML: Model, schemas, types, logics, and queries, in: Logics for Emerging Applications of Databases, 2003, pp. 1–41.
[15] A. Vakali, B. Catania, A. Madalena, XML data stores: emerging practices, IEEE Internet Computing 9 (2) (2005) 62–69.
[16] P. Aiken, M. Allen, XML in Data Management, Morgan Kaufmann, 2004.
[17] G. Powell, Beginning XML Databases, Wrox, 2006.
[18] Intellor-Group, Survey on XML database adoption signals profound changes in data management, http://www.softwareag.com/xml/library/intellor_database_survey.htm, 2008.
[19] M. Levene, G. Loizu, A Guided Tour of Relational Databases and Beyond, Springer, 1999.
[20] V. Apparao, et al., Document Object Model (DOM) level 1 specification, w3C recommendation, http://www.w3.org/REC-DOM-Level-1, 1998.

[21] A. Berglund, S. Boag, D. Chamberlin, M. Fernandez, M. Kay, J. Robe, J. Siméon, XML path language (XPath) 2.0, w3C Working Draft, http://www.w3.org/TR/xpath2.0/, 2005.
[22] S. Libkin, Elements of Finite Model Theory, Springer, 2004.
[23] D. Maier, The Theory of Relational Databases, Computer Science Press, 1983.
[24] M. Lee, T. Ling, W.L. Low, Designing functional dependencies for XML, in: 8th International Conference on Extending Database Technology, Springer, Prague, Czech Republic, 2002, pp. 124–141.
[25] T. Cover, J. Thomas, Elements of Information Theory, Wiley–Interscience, 2006.
[26] M.W. Vincent, Semantic foundations of 4NF in relational database design, Acta Inform. 36 (1999) 1–41.
[27] G. Gottlob, M. Schrefl, M. Stumptner, On the interaction between transitive closure and functional dependencies, in: 2nd Symposium on Mathematical Fundamentals of Database Systems, 1989, pp. 187–206.
[28] P. Buneman, S. Davidson, W. Fan, C. Hara, W. Tan, Keys for XML, Comput. Netw. 39 (5) (2002) 473–487.
[29] W. Fan, L. Libkin, On XML integrity constraints in the presence of DTDs, J. ACM 49 (3) (2002) 368–406.
[30] S. Hartmann, S. Link, T. Trinh, Efficient reasoning about XFDs with pre-image semantics, in: DASFAA, 2007, pp. 1070–1074.
[31] S. Hartmann, T. Trinh, Axiomatising functional dependencies for XML with frequencies, in: 4th International Symposium on Foundations of Information and Knowledge Systems (FoIKS), Springer, Budapest, Hungary, 2006, pp. 159–178.
[32] C. Yu, H. Jagadish, Efficient discovery of XML data redundancies, in: 32nd International Conference on Very Large Databases (VLDB), Morgan Kaufmann, 2006, pp. 103–114.
[33] Y. Chen, S. Davidson, C. Hara, Y. Zheng, RRXS:redundancy reducing XML storage in relations, in: 29th International Conference on Very Large Databases, Morgan Kaufmann, Berlin, Germany, 2003, pp. 189–200.
[34] K.-D. Schewe, Redundancy, dependencies and normal forms for XML databases, in: 16th Australasian Database Australian Database Conference, Australian Computer Society, Newcastle, Australia, 2005, pp. 7–16.
[35] J. Wang, R. Topor, Removing XML data redundancies using functional and equality-generating dependencies, in: 16th Australasian Database Conference, Australian Computer Society, Newcastle, Australia, 2005, pp. 65–74.
[36] W. Mok, On utilizing variables for specifying FDs in data-centric XML documents, Data & Knowledge Engineering 60 (3) (2007) 494–510.
[37] J. Wang, A comparative study of functional dependencies for XML, in: APWeb, 2005, pp. 308–319.
[38] M. Vincent, J. Liu, Checking functional dependency satisfaction in XML, in: 3rd International Symposium on Database and XML Technologies, Springer, Trondheim, Norway, 2005, pp. 4–17.
[39] S. Kolahi, L. Libkin, XML design for relational storage, in: 16th International World Wide Web Conference (WWW), 2007, pp. 114–223.
[40] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D.J. DeWitt, J.F. Naughton, Relational databases for querying XML documents: Limitations and opportunities, in: 25th International Conference on Very Large Databases (VLDB), Morgan Kaufmann, Edinburgh, Scotland, 1999, pp. 302–314.
[41] D. Florescu, D. Kossman, Storing and querying XML data using an RDBMS, IEEE Data Eng. Bull. 22 (3) (1999) 27–34.
[42] S. Kolahi, Dependency-preserving normalization of relational and XML data, in: 10th International Symposium on Database Programming Languages (DBPL), Springer, Trondheim, Norway, 2005, pp. 247–261.
[43] S. Kolahi, Dependency-preserving normalization of relational and XML data, J. Comput. System Sci. 73 (4) (2007) 636–647.
[44] M. Arenas, L. Libkin, An information-theoretic approach to normal forms for relational and XML data, in: 22nd ACM SIGACT-SIGMOD-SIGART Symposium of Principles of Database Systems (PODS), ACM, San Diego, CA, USA, 2003, pp. 15–26.
[45] A. Møller, M. Schwartzbach, Introduction to XML and Web Technologies, Addison–Wesley, 2006.
[46] W. Fan, XML publishing: Bridging theory and practice, in: Database Programming Languages, 2007, pp. 1–16.
[47] H.S. Thompson, D. Beech, M. Maloney, N. Mendelsohn, XML schema part 1: structures second edition, w3C Working Draft, http://www.w3.org/TR/xmlschema-1, 2004.
[48] T. Trinh, Using transversals for discovering XML functional dependencies, in: FoIKS, 2008, pp. 199–218.
[49] P. Buneman, S. Khanna, K.T.W. Tan, Archiving scientific data, ACM Trans. Database Syst. 29 (2004) 2–42.